Expediting Deep Learning with Transfer Learning: PyTorch Playbook

GETTING STARTED WITH TRANSFER LEARNING

Janani Ravi
CO-FOUNDER, LOONYCORN
www.loonycorn.com

Overview

Understand the use of pre-trained models and transfer learning

Understand source and destination domains

Understanding source and destination tasks

Learn when to use transfer learning

Explore PyTorch support for transfer learning

Prerequisites and Course Outline

Prerequisites

Comfortable programming in Python

Basic understanding of neural networks

Worked with PyTorch to build and train neural networks

Prerequisite Courses

Foundations of PyTorch

Building Your First PyTorch Solution

Image Classification With PyTorch

Course Outline

Understanding and leveraging transfer learning

Performing fixed feature extraction with pre-trained models

Reusing model architectures and designs

Transfer Learning

Transfer Learning

The practice of re-using a trained neural network that solves a problem similar to yours, usually leaving the network architecture unchanged and re-using some or all of the model weights.

Transfer Learning

The practice of re-using a trained neural network that solves a problem similar to yours, usually leaving the network architecture unchanged and re-using some or all of the model weights.

Avoid designing NN architecture from scratch

Transfer Learning

The practice of re-using a trained neural network that solves a problem similar to yours, usually leaving the network architecture unchanged and re-using some or all of the model weights.

Only makes sense for common, widely studied use-cases

Transfer Learning

The practice of re-using a trained neural network that solves a problem similar to yours, usually leaving the network architecture unchanged and re-using some or all of the model weights.

In which basic problem structure stays same, but details vary

Transfer Learning

The practice of re-using a trained neural network that solves a problem similar to yours, usually leaving the network architecture unchanged and re-using some or all of the model weights.

Image recognition, language translation are classic examples

Transfer Learning

The practice of re-using a trained neural network that solves a problem similar to yours, usually leaving the network architecture unchanged and re-using some or all of the model weights.

Often the hardest part - allows us to “stand on the shoulders of giants”

Transfer Learning

The practice of re-using a trained neural network that solves a problem similar to yours, usually leaving the network architecture unchanged and re-using some or all of the model weights.

Re-train from scratch, fine-tune model weights, use entirely as-is

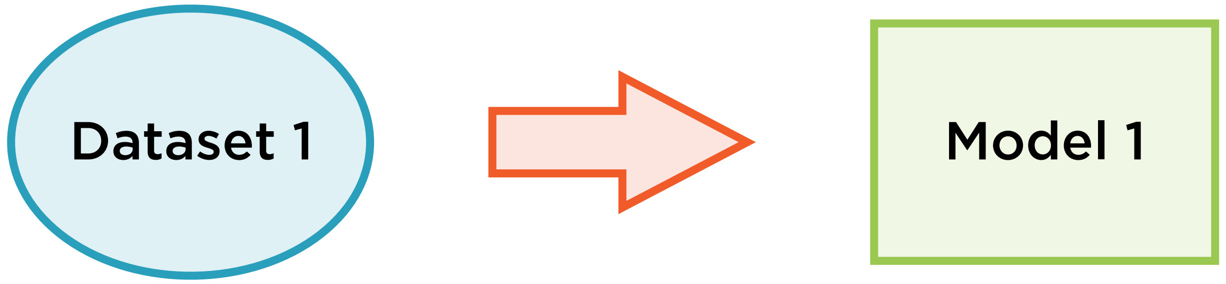
Transfer Learning

The practice of re-using a trained neural network that solves a problem similar to yours, usually leaving the network architecture unchanged and re-using some or all of the model weights.

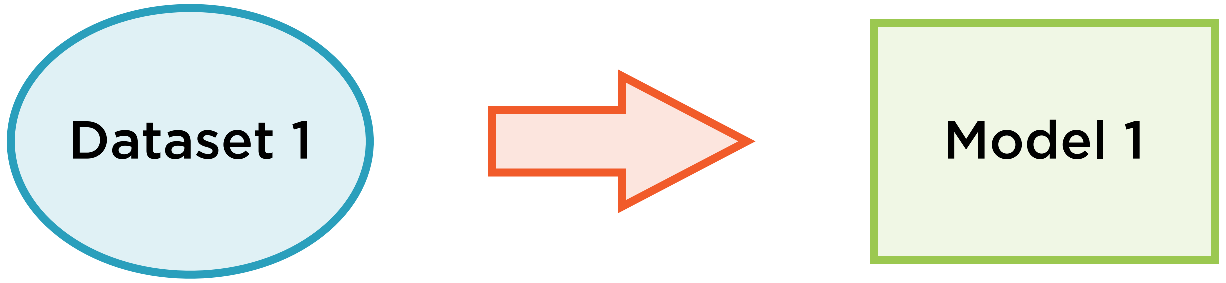
Several choices based on size and similarity of datasets

Traditional ML vs. Transfer Learning

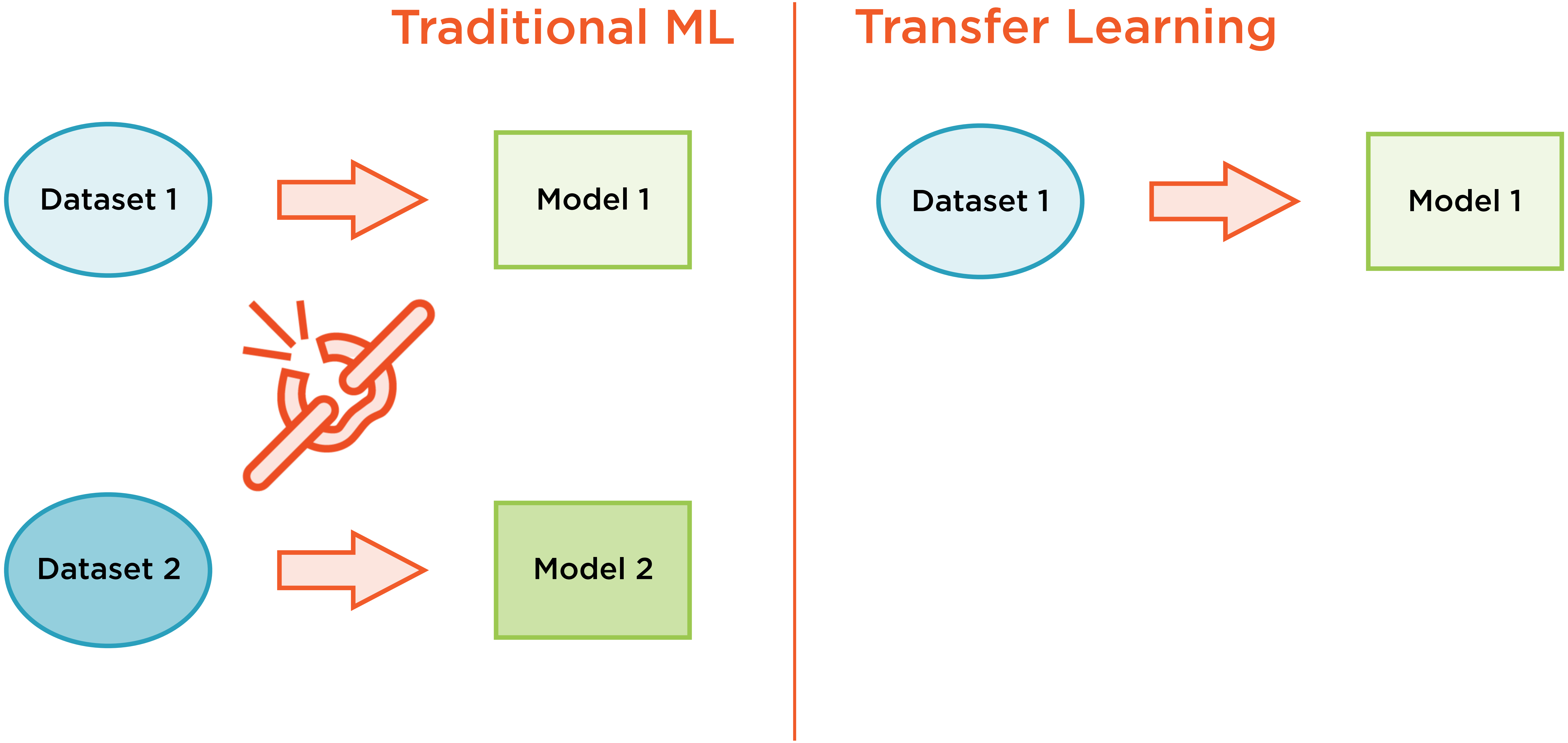
Traditional ML



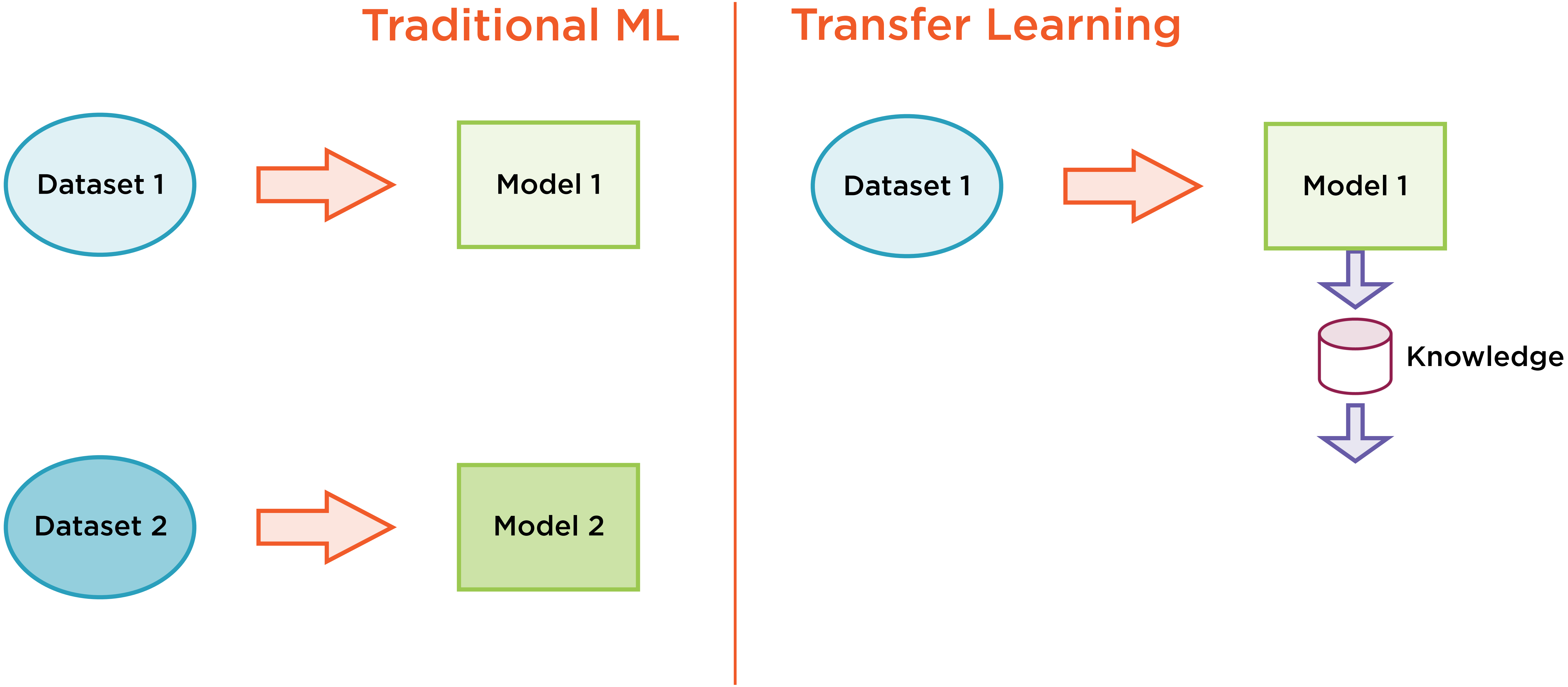
Transfer Learning



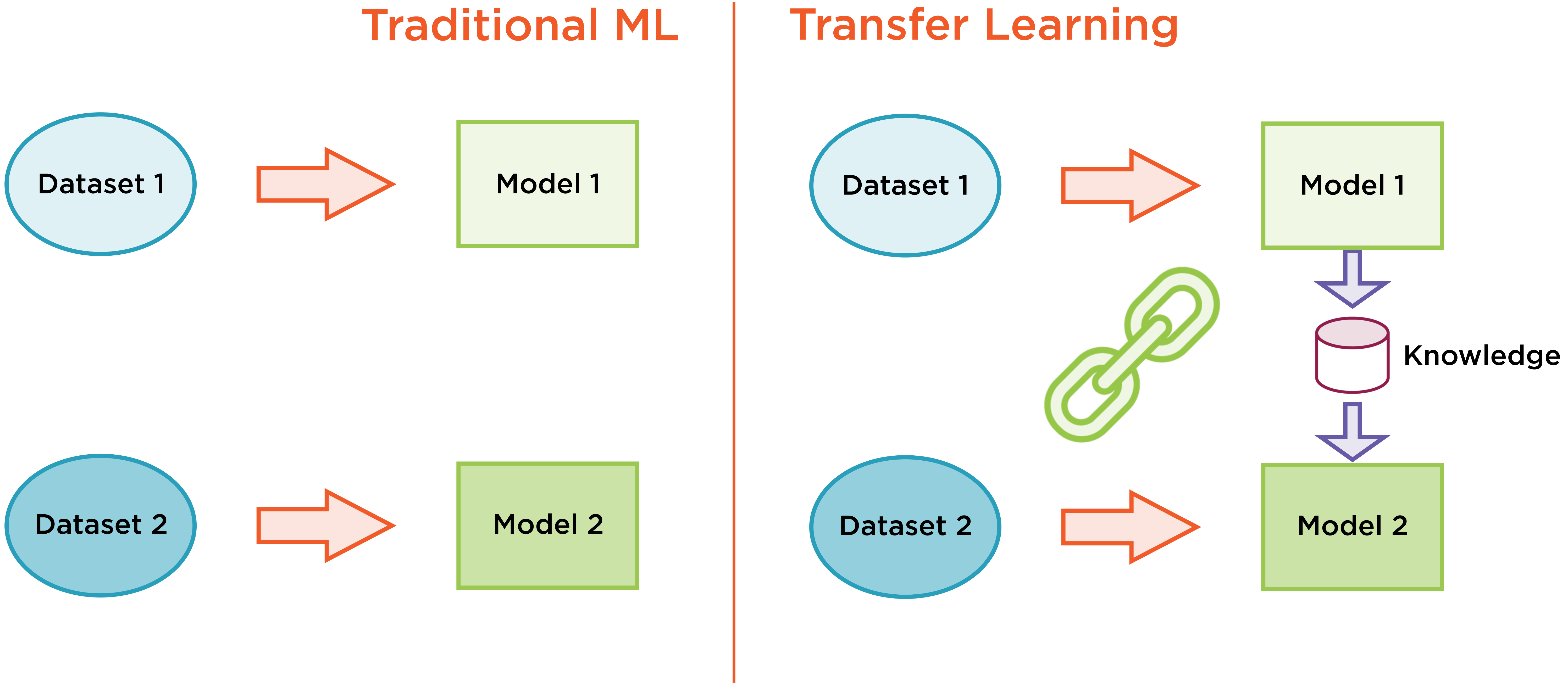
Traditional ML vs. Transfer Learning



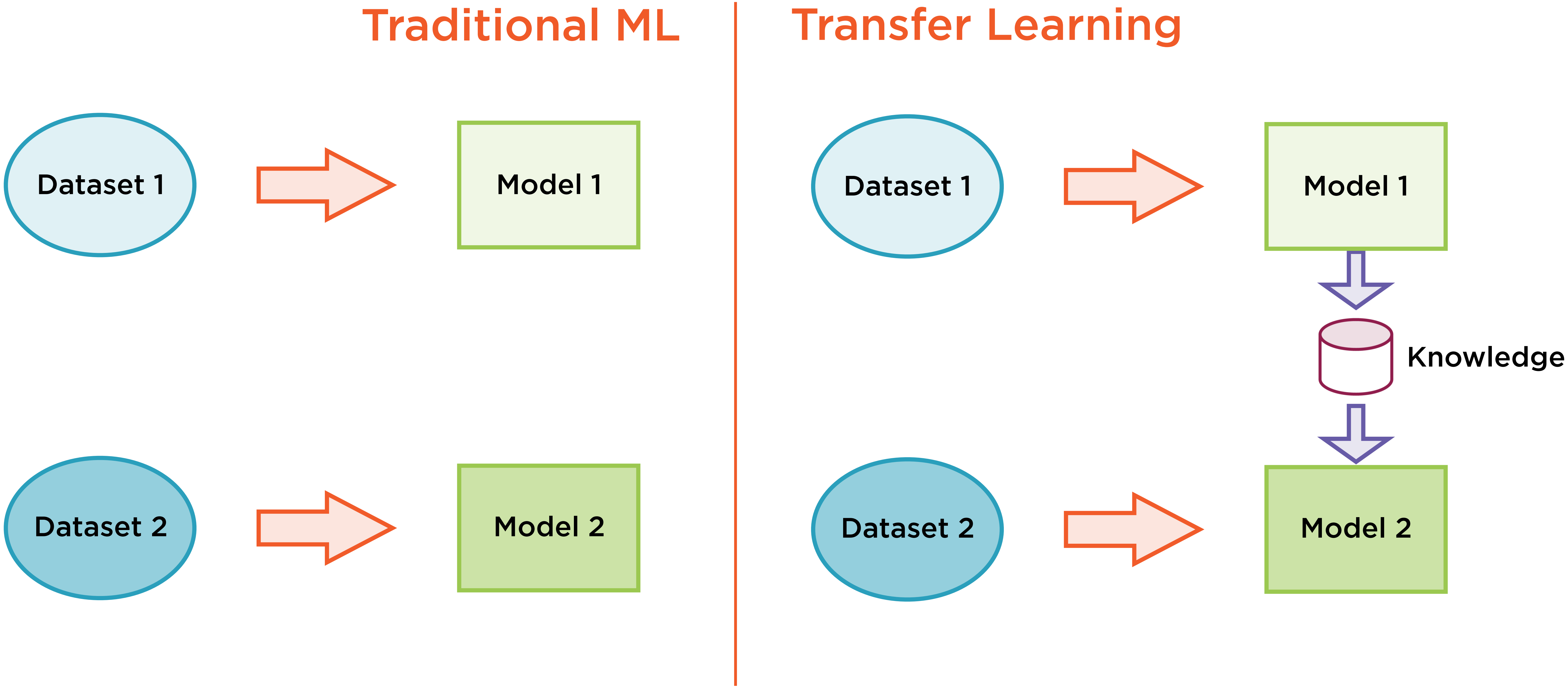
Traditional ML vs. Transfer Learning



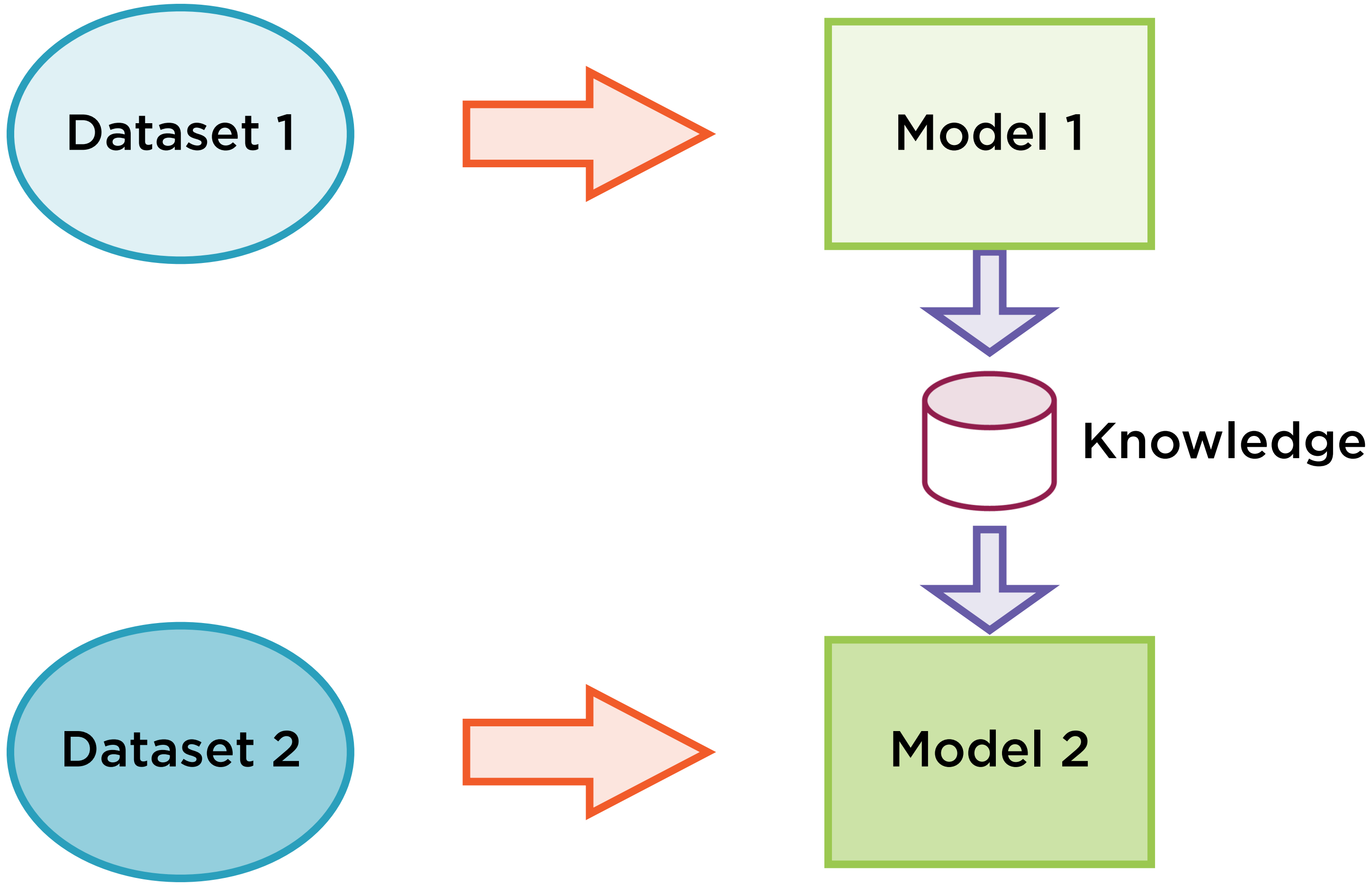
Traditional ML vs. Transfer Learning



Traditional ML vs. Transfer Learning



Transfer Learning



Dataset 1

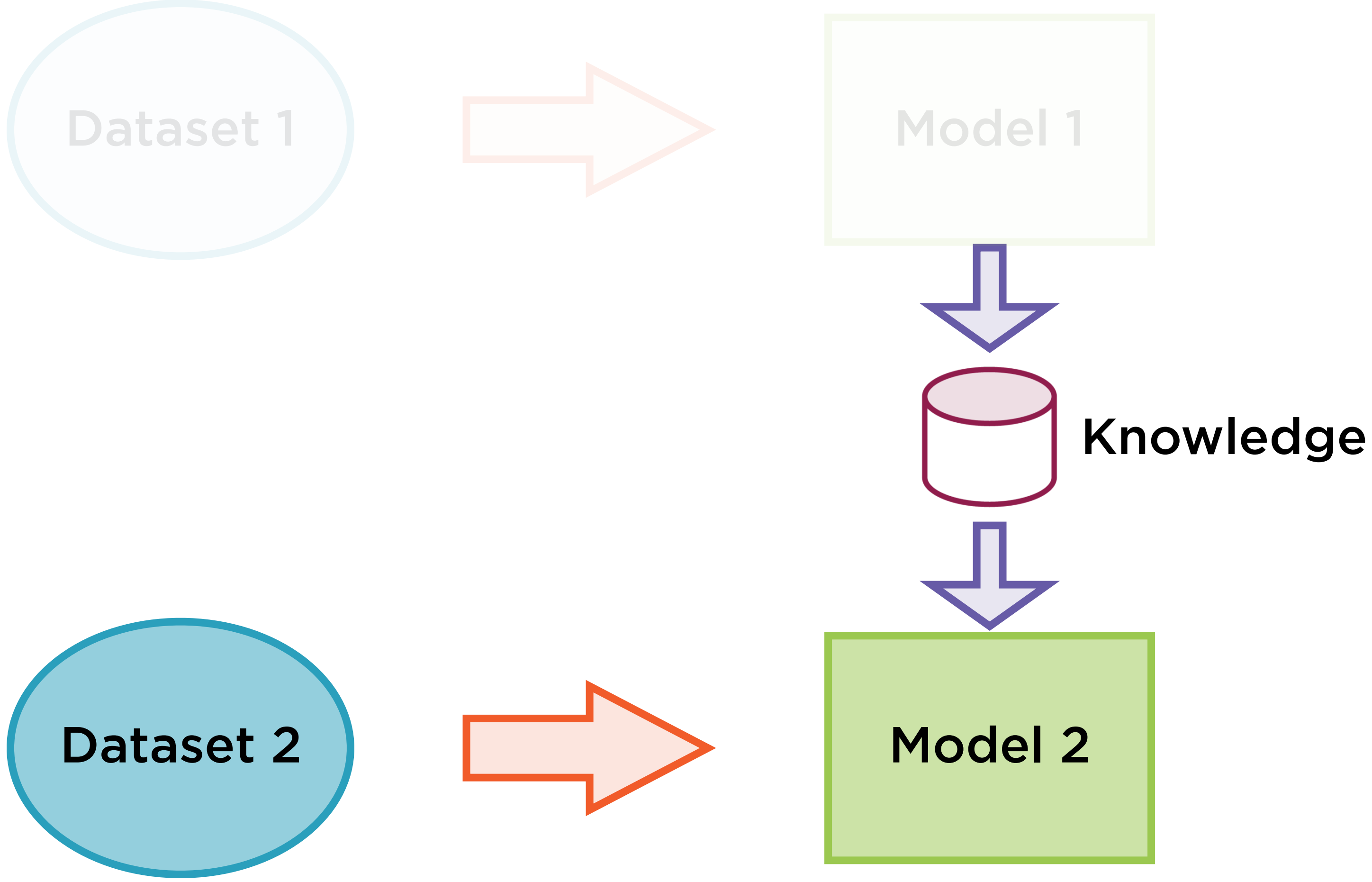
Model 1

Knowledge

Dataset 2

Model 2

Transfer Learning



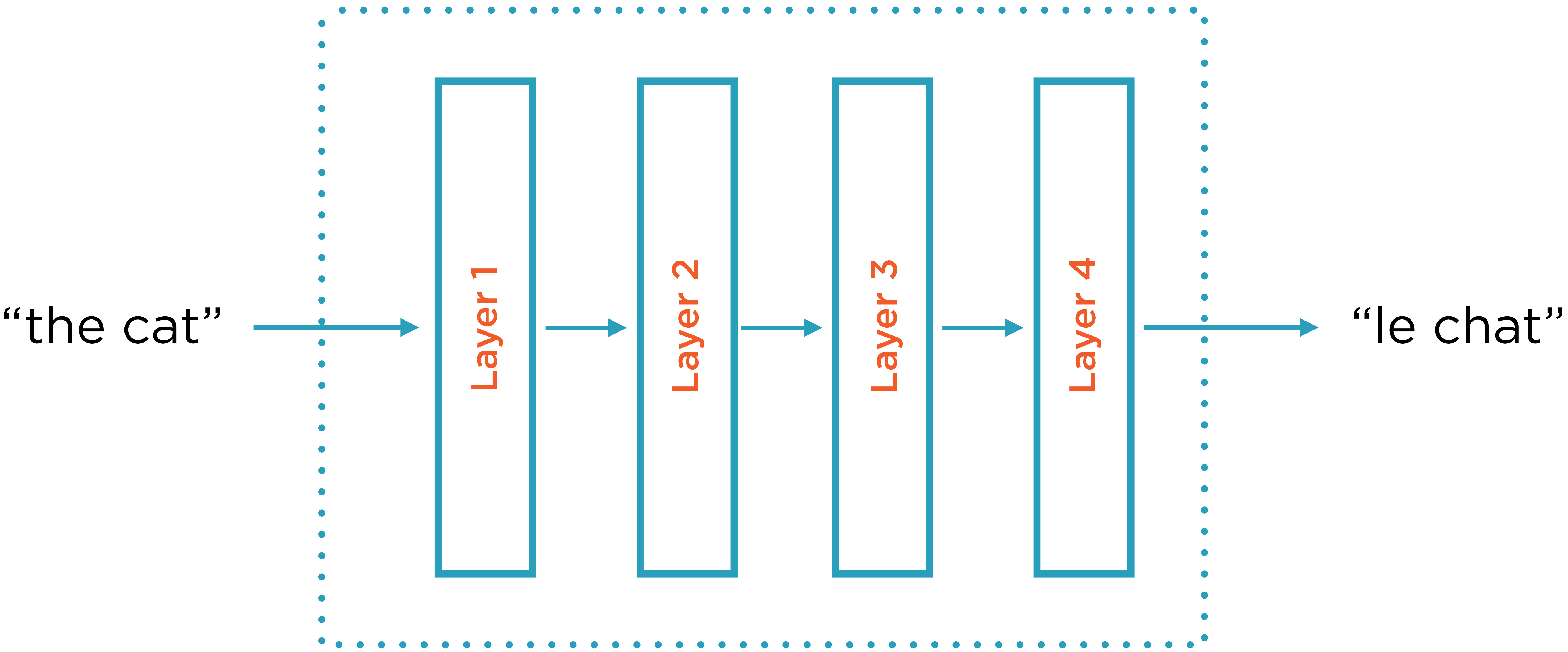
Transferred knowledge is especially useful when the new dataset is small and not sufficient to train a model from scratch

Source and Target Domains and Tasks

A Survey on Transfer Learning

https://ieeexplore.ieee.org/document/5288526

Original Model: English to French

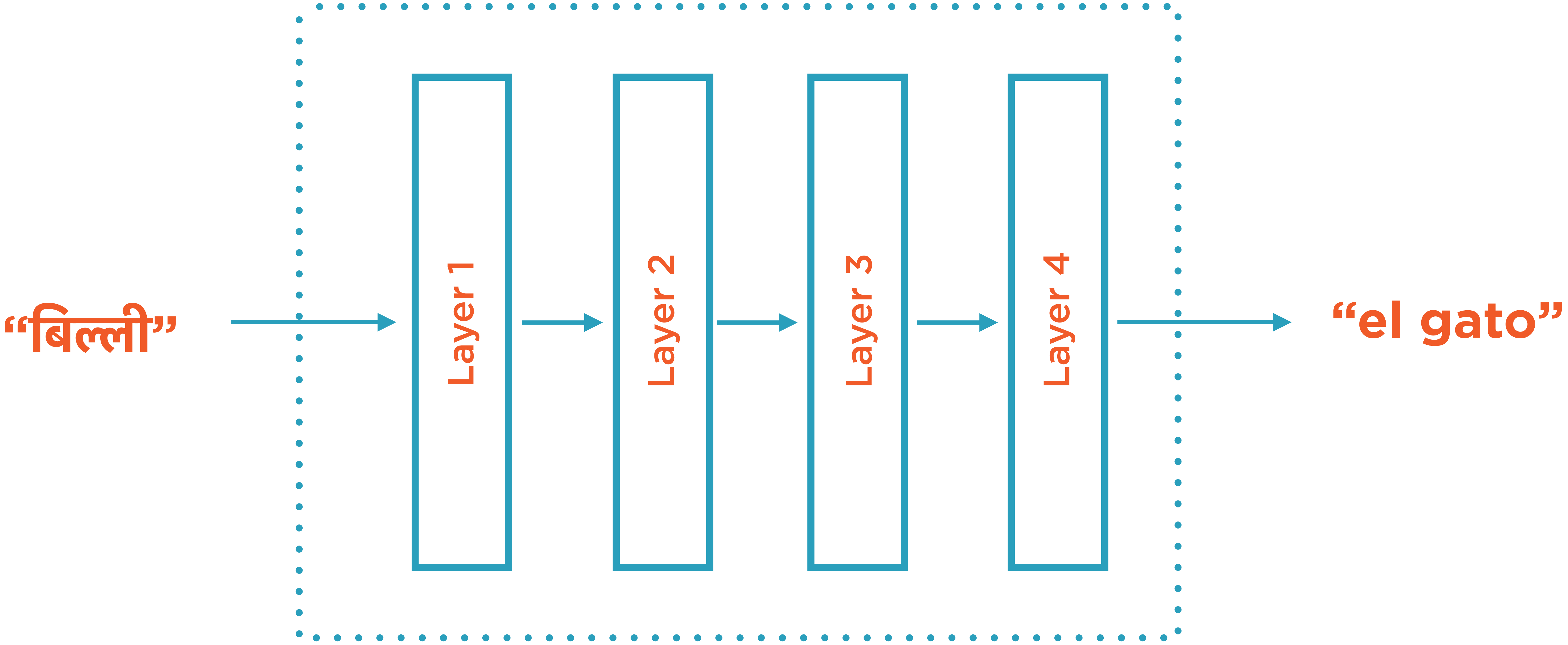


English input

Highly optimized NN architecture

French output

Transfer Learning: Hindi to Spanish



Hindi input

Highly optimized NN architecture

Spanish output

Source and Target Domains

Source Domain

English

Target Domain

Hindi

Domains refer to where the X variables are drawn from and how they are distributed

Source and Target Tasks

Source Task

Translation to French

Target Task

Translation to Spanish

Tasks refer to the Y variables
and how they are related to X

Source and Target Domains

Target Domain:
Labels available?

Source Domain:
Labels available?

Source and Target Domains

Target Domain:
Labels available?

Source Domain:
Labels available?

		Yes
		No

Source and Target Domains

Target Domain:
Labels available?

Source Domain:
Labels available?

Yes

No

Source and Target Domains

Target Domain:
Labels available?

Source Domain:
Labels available?

	Yes	No
Yes		
No		

Source and Target Domains

Target Domain: Labels available?

Source Domain: Labels available?

	Yes	No
Yes		
No		- Unsupervised transfer learning - Clustering - Dimensionality Reduction

Source and target domains and tasks are **different** but **related**

Source and Target Domains

Source and target domains are the **same**

Source and target tasks are **different** but **related**

Target Domain: Labels available?

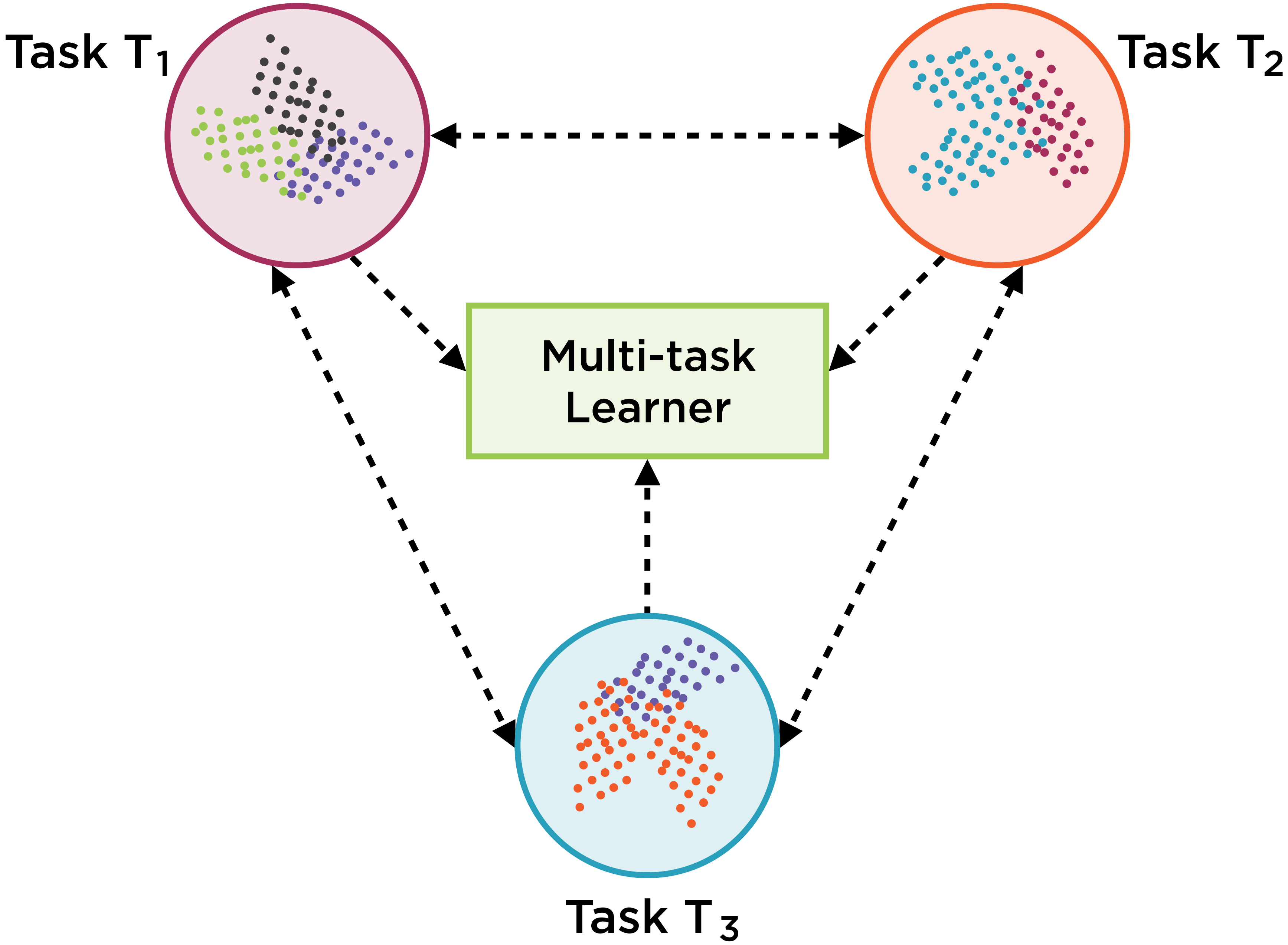
Source Domain: Labels available?

Source Domain: Labels available?	Target Domain: Yes	Target Domain: No
Yes	• Multi-task learning • Classification • Regression	
No		• Unsupervised transfer learning • Clustering • Dimensionality Reduction

Multi-task Learning

Subfield of machine learning in which multiple learning tasks are solved at the same time to exploit commonalities across tasks

Multi-task Learning



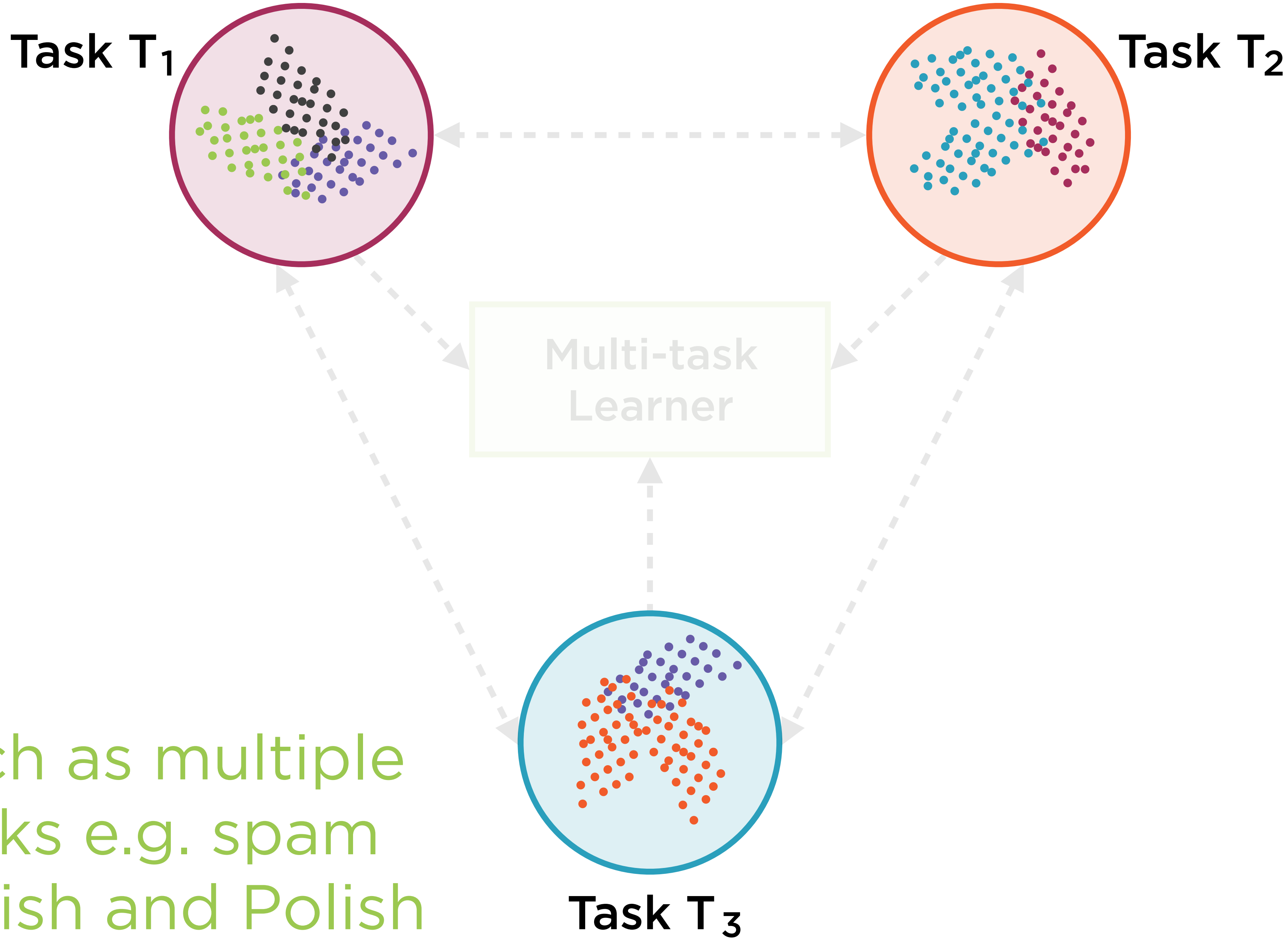
Task T_1

Task T_2

Multi-task Learner

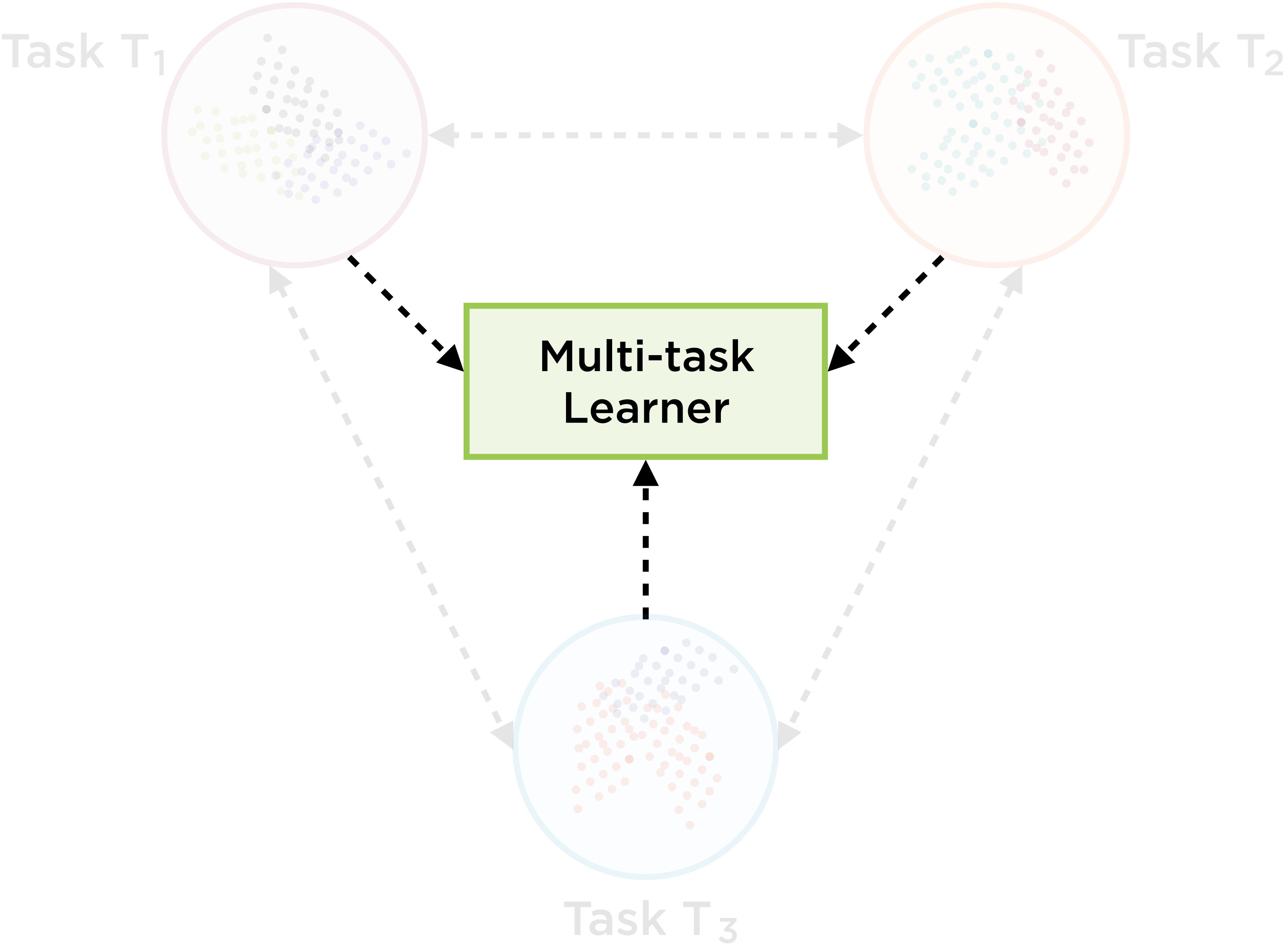
Task T_3

Multi-task Learning

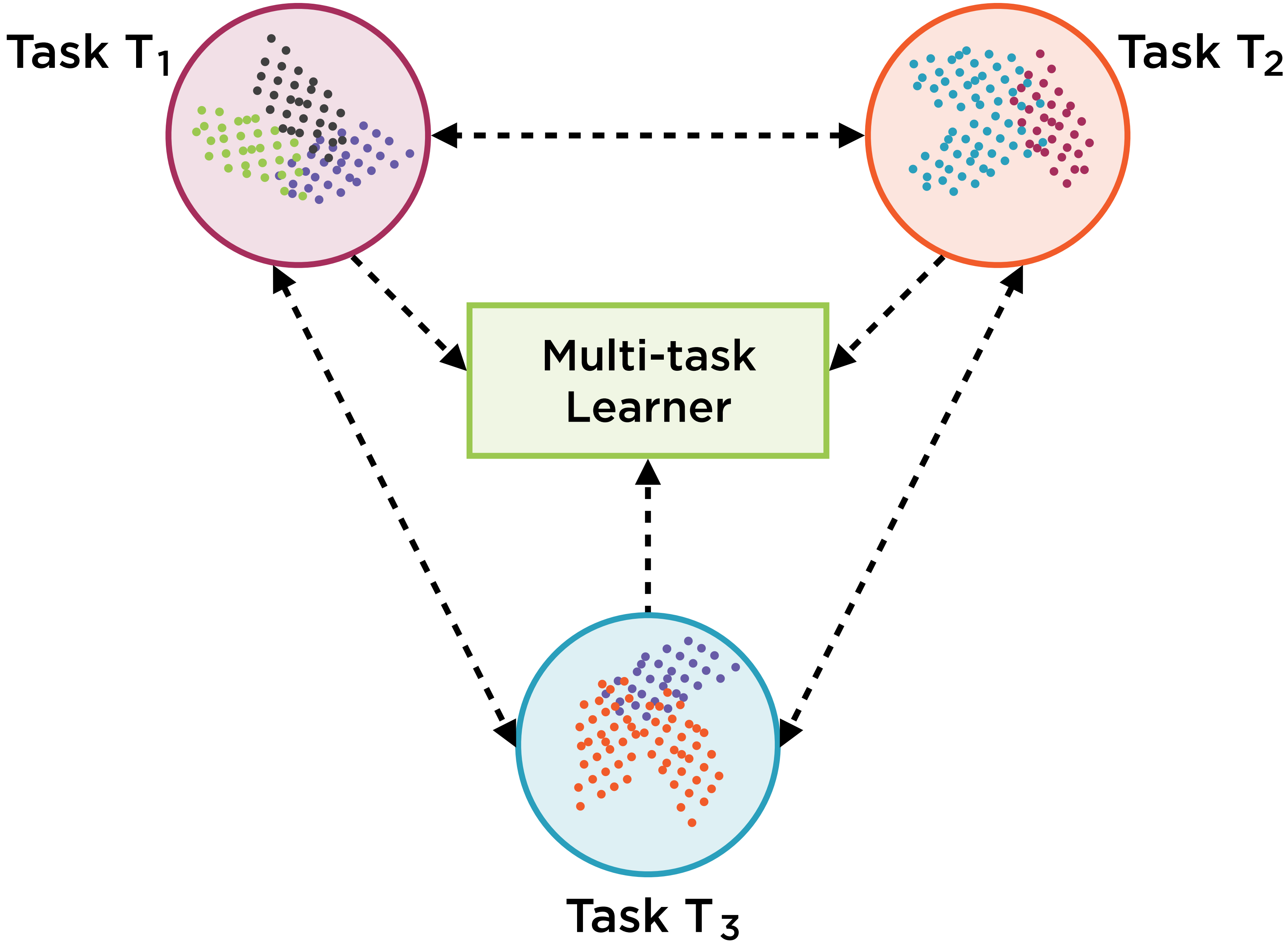


Multiple tasks such as multiple classification tasks e.g. spam detection for English and Polish language users

Multi-task Learning



Multi-task Learning



Source and Target Domains

Target Domain: Labels available?

Source Domain: Labels available?

	Yes	No
Yes	- Multi-task learning - Classification - Regression	
No		- Unsupervised transfer learning - Clustering - Dimensionality Reduction

Source and Target Domains

Target Domain: Labels available?

Source Domain: Labels available?

	Yes	No
Yes	- Multi-task learning - Classification - Regression	- Tricky - need expert judgment - Are source and target domains the same?
No		- Unsupervised transfer learning - Clustering - Dimensionality Reduction

Source and Target Domains

Target Domain:
Labels available?

Source Domain:
Labels available?

	Yes	No
Yes	• Multi-task learning • Classification • Regression	• Yes - Domain is the same fix biases • No - Domain is different, needs domain adaptation
No		• Unsupervised transfer learning • Clustering • Dimensionality Reduction

Source and target domains **different** but **related**

Source and target tasks are the **same**

Source and Target Domains

Target Domain: Labels available?

Source Domain: Labels available?

	Yes	No	
	- Multi-task learning - Classification - Regression	- Yes - Domain is the same fix biases - No - Domain is different, needs domain adaptation	Yes
	- Self-taught learning - Transfer learning from unlabeled data - Widely used in practice	- Unsupervised transfer learning - Clustering - Dimensionality Reduction	No

Source and target domains **different** but **related**

Source and target tasks are the **same**

Self-taught Learning

Can be thought of as semi-supervised, transfer learning. Uses labeled data belonging to the desired classes and unlabeled data from other similar classes.

Self-taught Learning

Can be thought of as semi-supervised, transfer learning. Uses labeled data belonging to the desired classes and unlabeled data from other similar classes.

Source and Target Domains

Target Domain: Labels available?

Source Domain: Labels available?	Yes	No
Yes	• Multi-task learning • Classification • Regression	• Yes - Domain is the same fix biases • No - Domain is different, needs domain adaptation
No	• Self-taught learning • Transfer learning from unlabeled data • Widely used in practice	• Unsupervised transfer learning • Clustering • Dimensionality Reduction

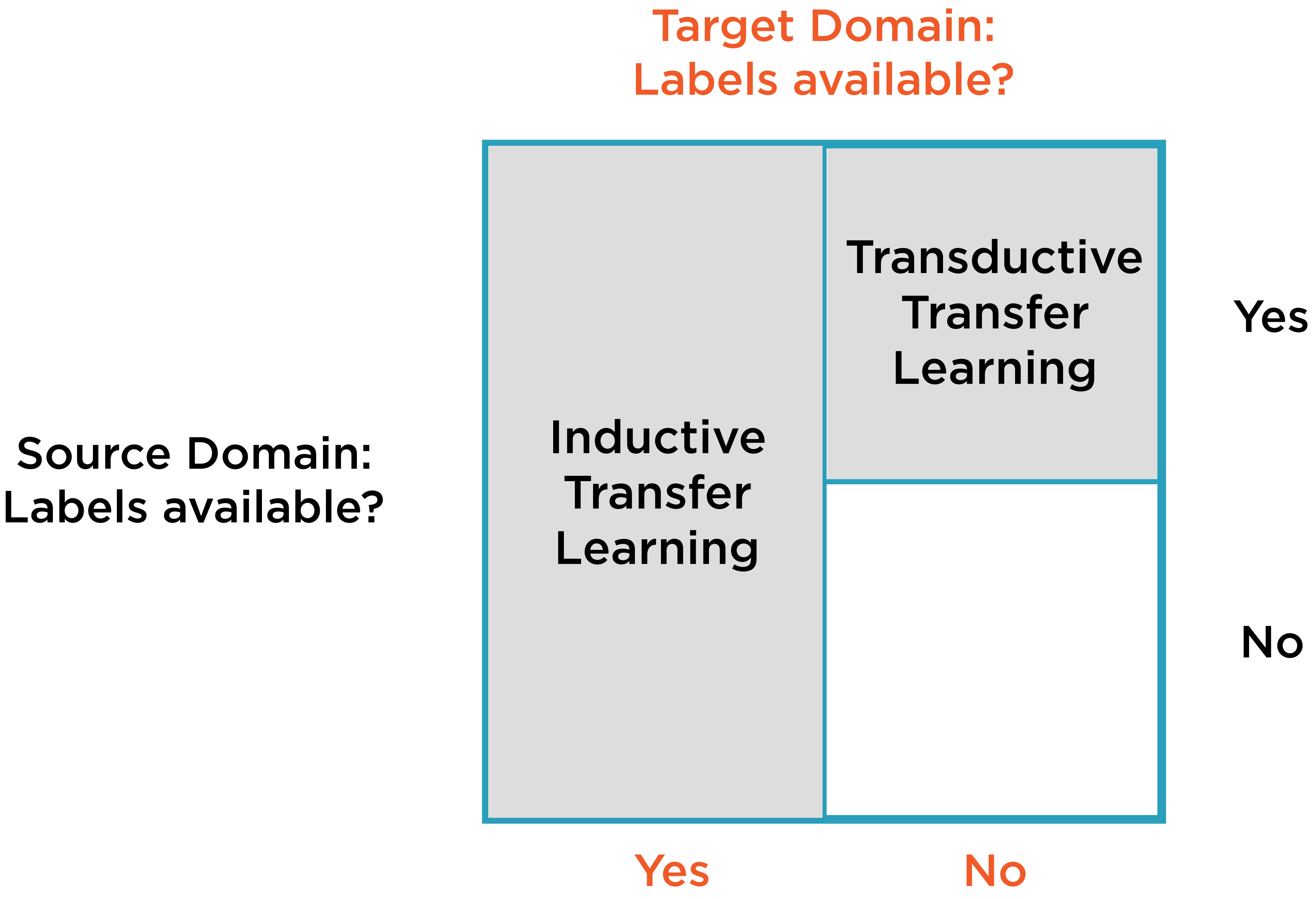
Source and Target Domains

Target Domain:
Labels available?

Source Domain:
Labels available?

	Yes	No
Yes	Inductive Transfer Learning	
No		

Source and Target Domains



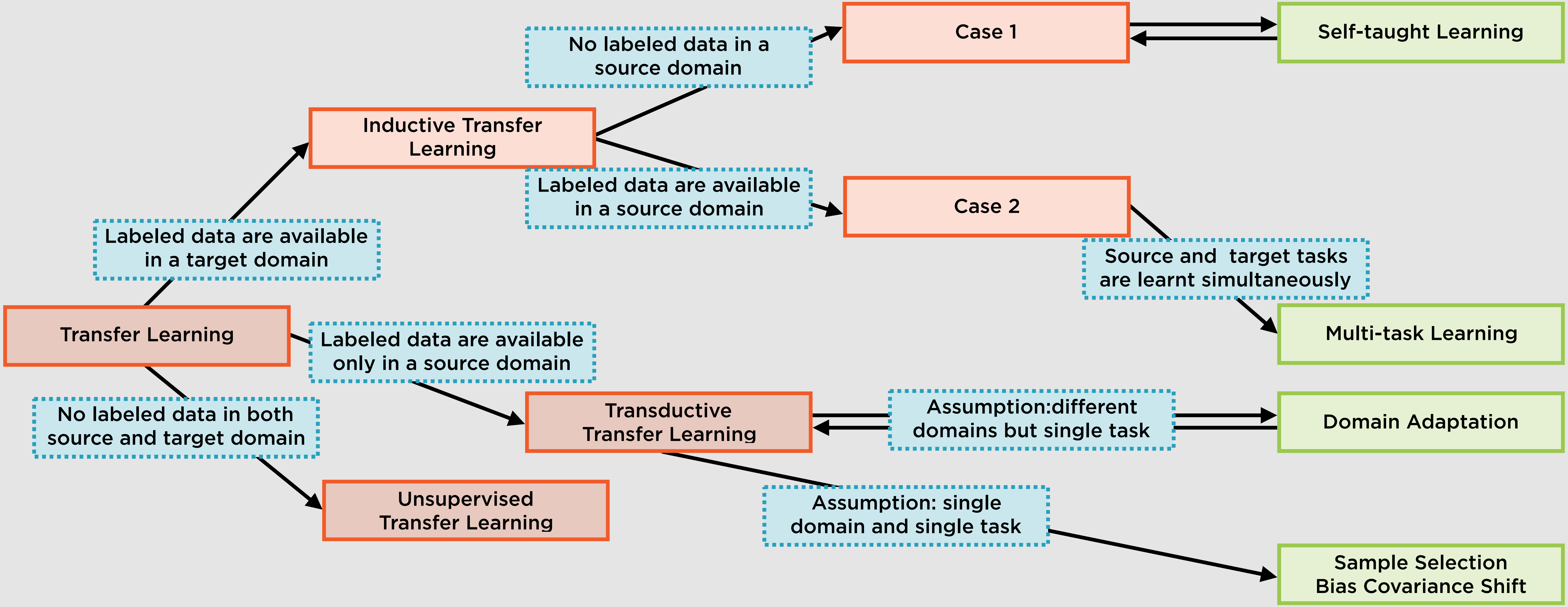
Source Domain: Labels available?	Target Domain: Labels available? Yes	Target Domain: Labels available? No
Yes	Inductive Transfer Learning	Transductive Transfer Learning
No		

Source and Target Domains

Target Domain:
Labels available?

Source Domain: Labels available?	Yes	No
Yes	Inductive Transfer Learning	Transductive Transfer Learning
No		Unsupervised Transfer Learning

Transfer Learning Strategies



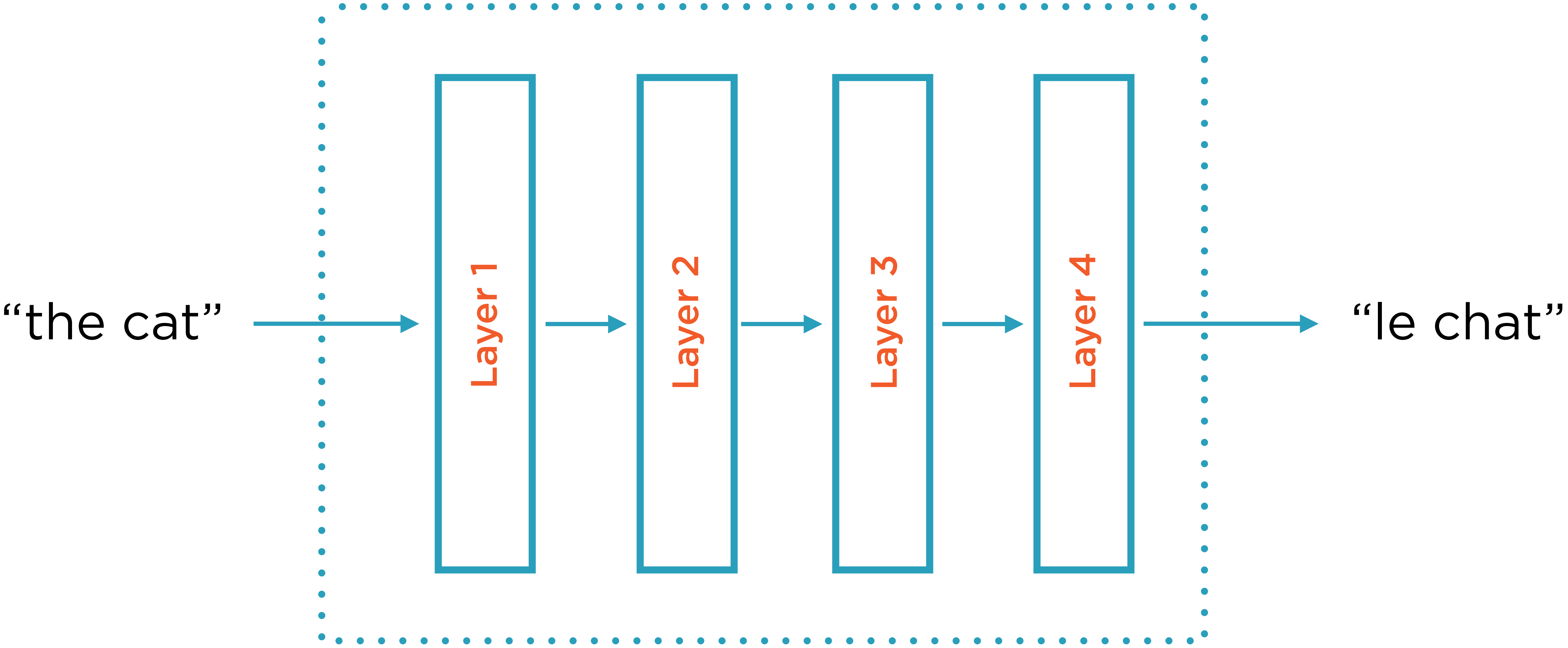
Transfer Learning
Labeled data are available in a target domain
Inductive Transfer Learning
No labeled data in a source domain
Case 1
Self-taught Learning
Labeled data are available in a source domain
Case 2
Source and target tasks are learnt simultaneously
Multi-task Learning
Labeled data are available only in a source domain
Transductive Transfer Learning
Assumption:different domains but single task
Domain Adaptation
Assumption: single domain and single task
Sample Selection Bias Covariance Shift
No labeled data in both source and target domain
Unsupervised Transfer Learning

Types of Transfer Learning Strategies and their Settings

Learning Strategy	Related Areas	Source & Target Domains	Sure Domain Labels	Target Domain Labels	Sourece & Target Tasks	Tasks
Inductive Transfer Learning	Multi_task Learning	The Same	Available	Available	Different but Related	Regression Classification
	Self-taught Learning	The Same	Unavailable	Available	Different but Related	Regression Classification
Unsupervised Transfer Learning		Different but Related	Unavailable	Unavailable	Different but Related	Clustering Dimensionality Reduction
Transductive Transfer Learning	Domain Adaptation,Sample Selection Bias & Co-variate Shift	Different but Related	Available	Unavailable	The Same	Regression Classification

Scenarios in Transfer Learning

Original Model: English to French

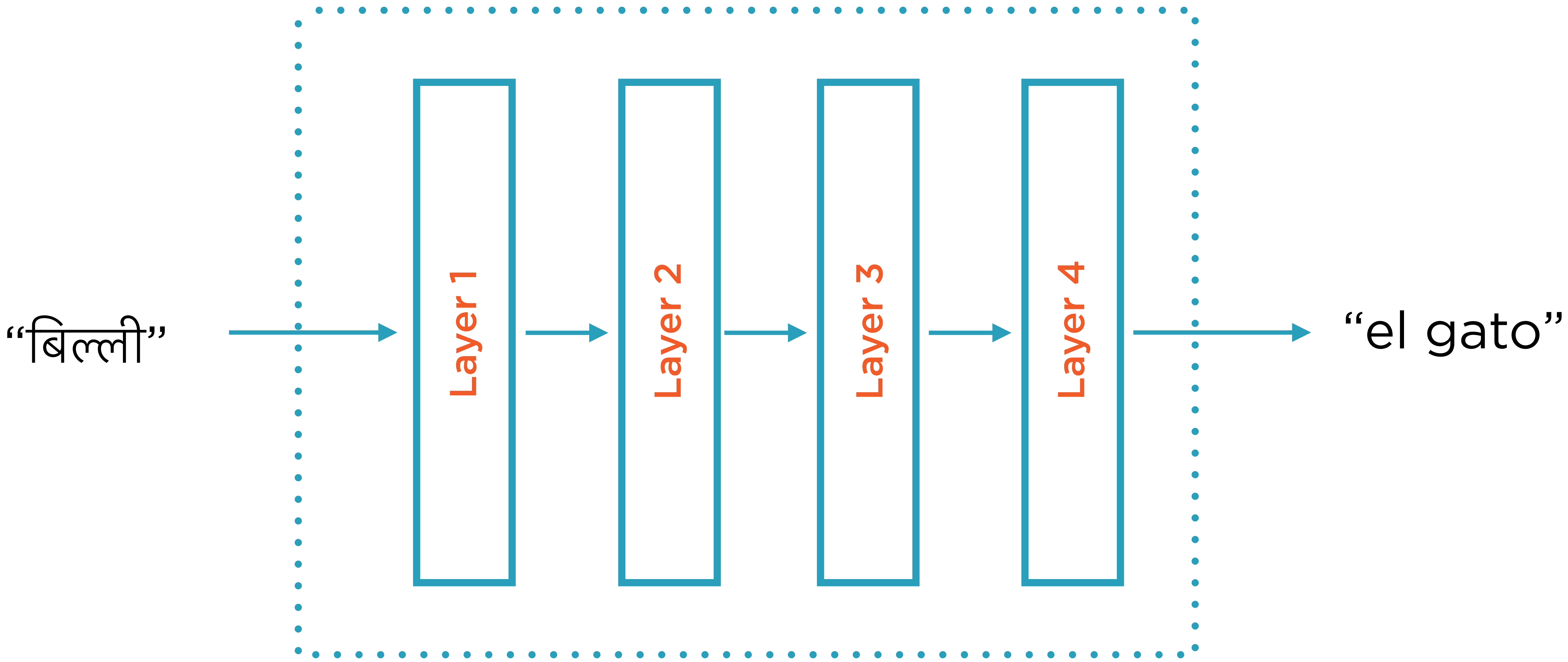


English input

Highly optimized NN architecture

French output

Transfer Learning: Hindi to Spanish



Hindi input

Highly optimized NN architecture

Spanish output

Re-training vs. Fine-tuning

Re-train from scratch

Find new model weights starting from scratch

Keep model architecture as-is

Fine-tune model weights

Find new model weights starting from original model weights

Keep model architecture as-is

Transfer Learning Scenarios

How similar are the old and new datasets?

How much new training data is available?

	Different	Similar
Lots		
Little		

Transfer Learning Scenarios

How similar are the old and new datasets?

How much new training data is available?

	Different	Similar
Lots	- Re-use architecture - Re-calculate model weights - Re-train from scratch	
Little		

Transfer Learning Scenarios

How similar are the old and new datasets?

How much new training data is available?

Different	Similar	
• Re-use architecture • Re-calculate model weights • Re-train from scratch		Lots
	• Re-use architecture • Fine-tune model weights • Fine-tune only final layers	Little

Transfer Learning Scenarios

How similar are the old and new datasets?

How much new training data is available?

	Different	Similar
Lots	- Re-use architecture - Re-calculate model weights - Re-train from scratch	- Re-use architecture - Fine-tune model weights - Fine-tune all layers
Little		- Re-use architecture - Fine-tune model weights - Fine-tune only final layers

Transfer Learning Scenarios

How similar are the old and new datasets?

How much new training data is available?

	Different	Similar
Lots	• Re-use architecture • Re-calculate model weights • Re-train from scratch	• Re-use architecture • Fine-tune model weights • Fine-tune all layers
Little	• Re-use architecture • Fine-tune model weights • Fit classifier on initial layers	• Re-use architecture • Fine-tune model weights • Fine-tune only final layers

Transfer Learning Scenarios

How similar are the old and new datasets?

How much new training data is available?

	Different	Similar
Lots	• Re-use architecture • Re-calculate model weights • Re-train from scratch	• Re-use architecture • Fine-tune model weights • Fine-tune all layers
Little	• Re-use architecture • Fine-tune model weights • Fit classifier on initial layers	• Re-use architecture • Fine-tune model weights • Fine-tune only final layers

Transfer Learning Scenarios

How similar are the old and new datasets?

How much new training data is available?

	Different	Similar
Lots	• Re-use architecture • Re-calculate model weights • Re-train from scratch	• Re-use architecture • Fine-tune model weights • Fine-tune all layers
Little	• Re-use architecture • Fine-tune model weights • Fit classifier on initial layers	• Re-use architecture • Fine-tune model weights • Fine-tune only final layers

Transfer Learning Scenarios

How similar are the old and new datasets?

How much new training data is available?

	Different	Similar
Lots	- Re-use architecture - **Re-calculate model weights** - Re-train from scratch	- Re-use architecture - **Fine-tune model weights** - Fine-tune all layers
Little	- Re-use architecture - **Fine-tune model weights** - Fit classifier on initial layers	- Re-use architecture - **Fine-tune model weights** - Fine-tune only final layers

Transfer Learning Scenarios

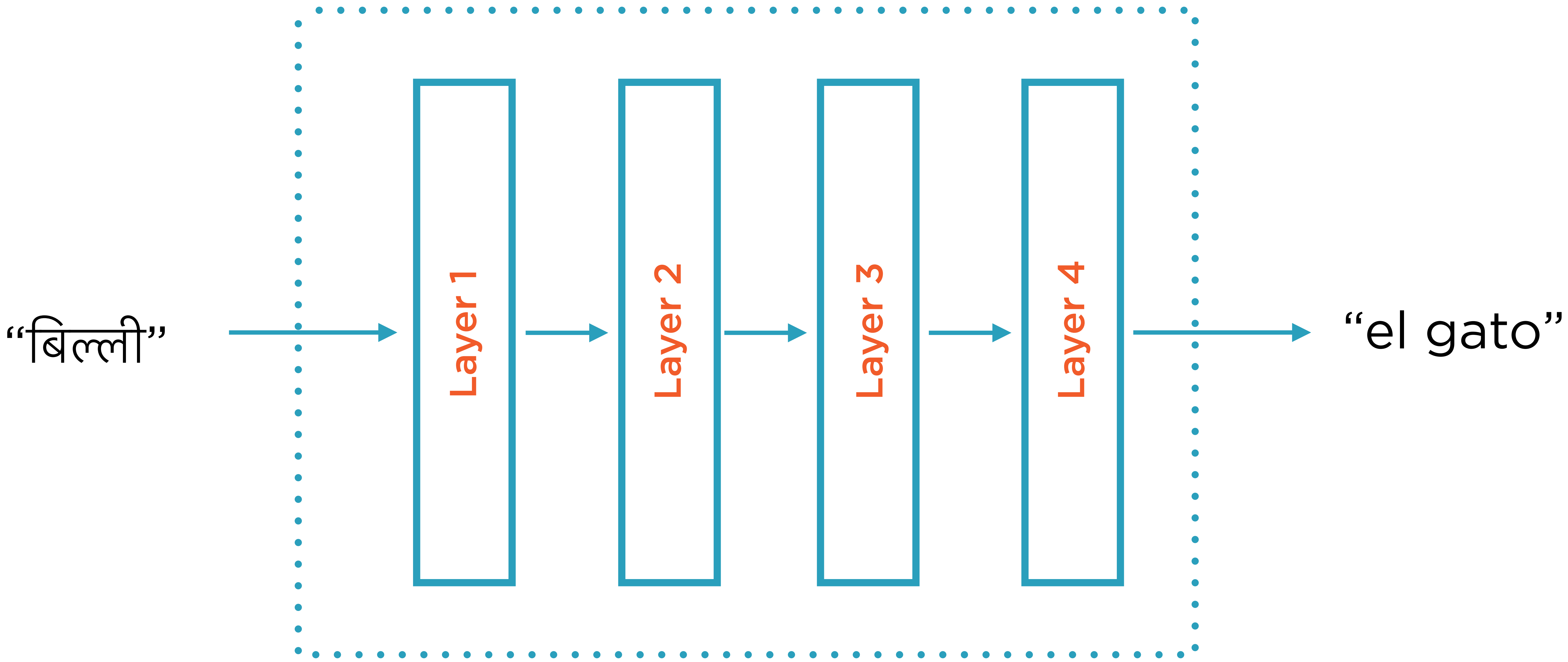
How similar are the old and new datasets?

How much new training data is available?

	Different	Similar
Lots	• Re-use architecture • Re-calculate model weights • Re-train from scratch	• Re-use architecture • Fine-tune model weights • Fine-tune all layers
Little	• Re-use architecture • Fine-tune model weights • Fit classifier on initial layers	• Re-use architecture • Fine-tune model weights • Fine-tune only final layers

Freeze or Fine-tune Layers

Transfer Learning: Hindi to Spanish

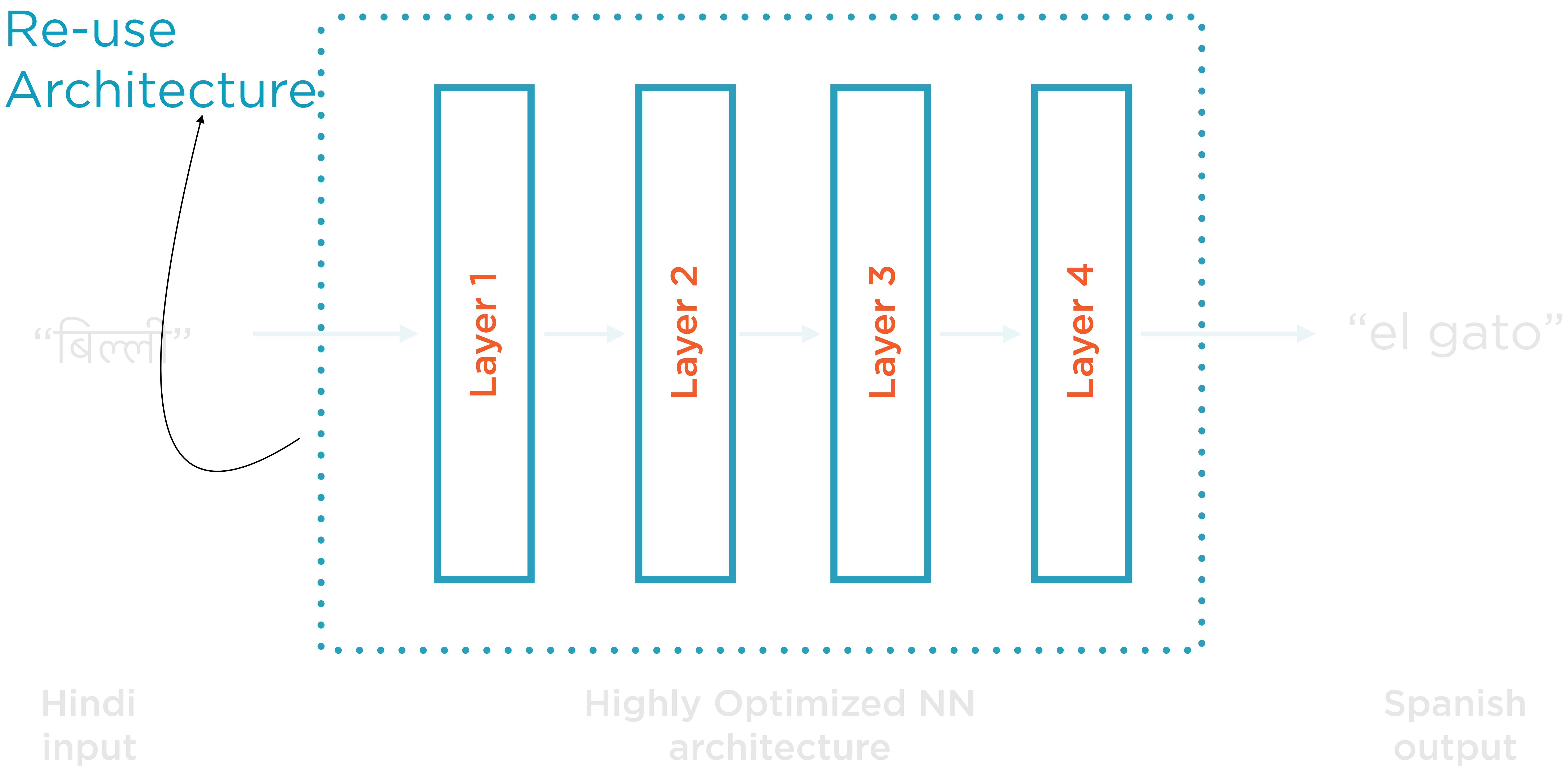


Hindi input

Highly optimized NN architecture

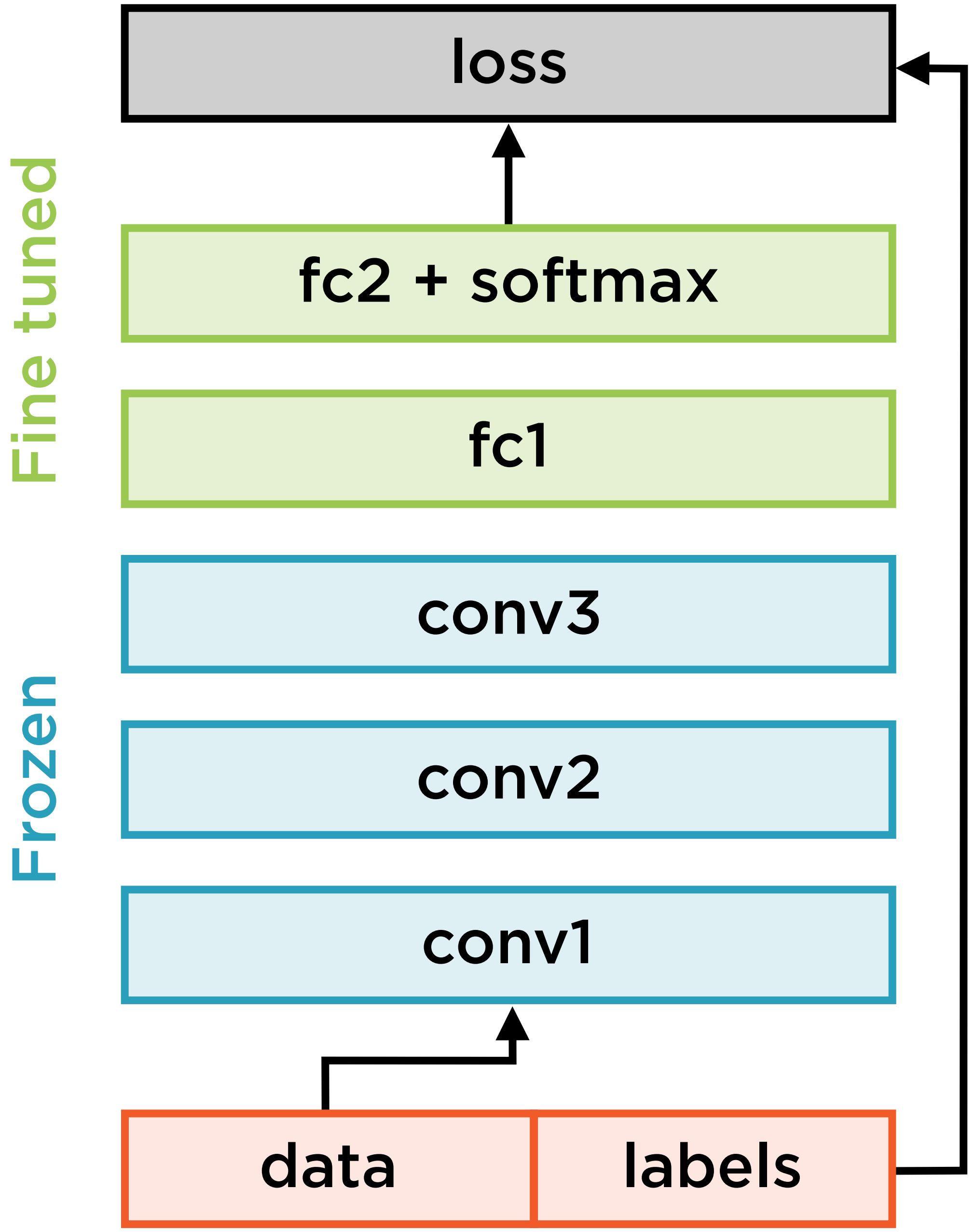
Spanish output

Transfer Learning: Hindi to Spanish

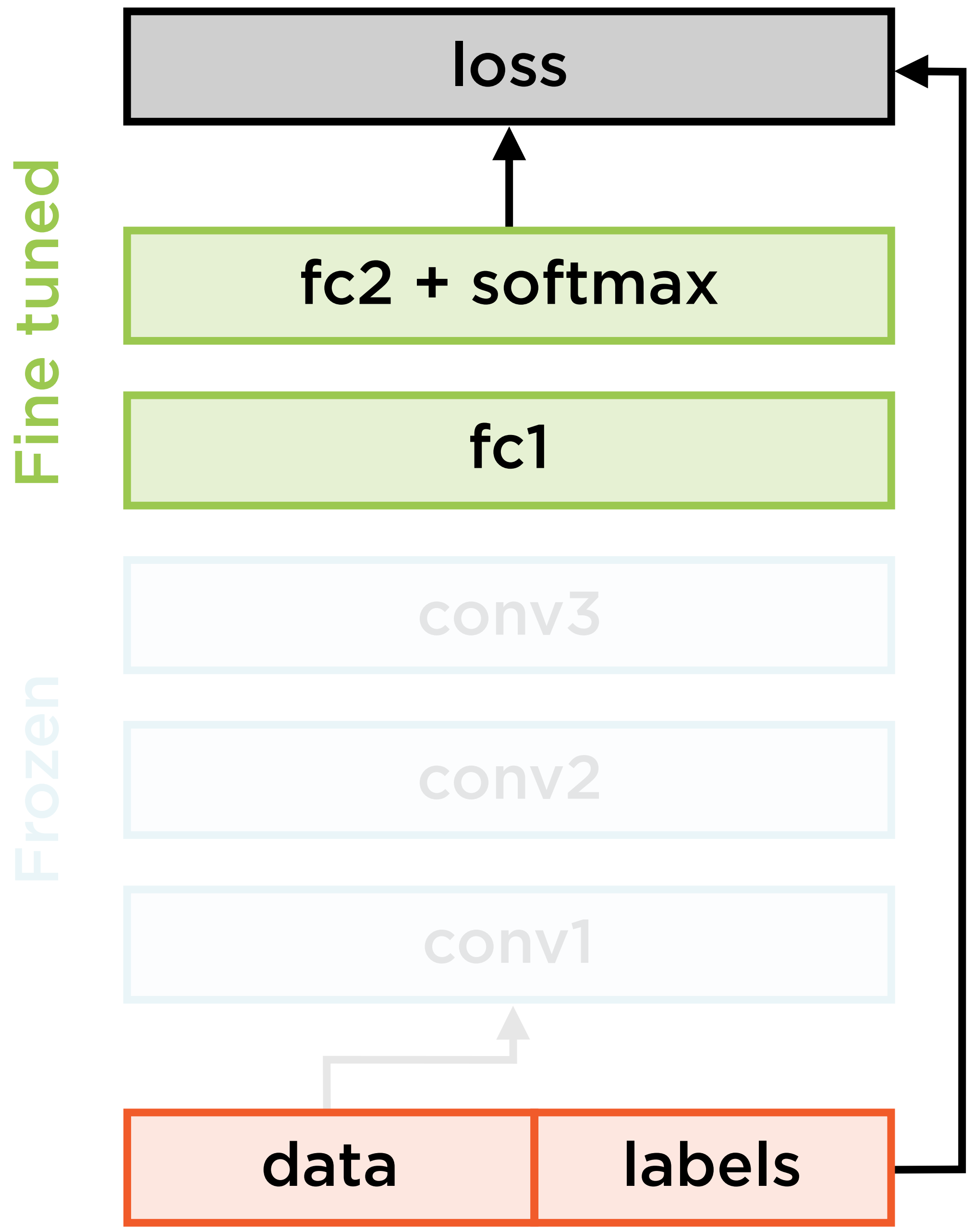


Usually the **top** (later) layers of the neural network are **more specific** to the problem and will need to be tuned

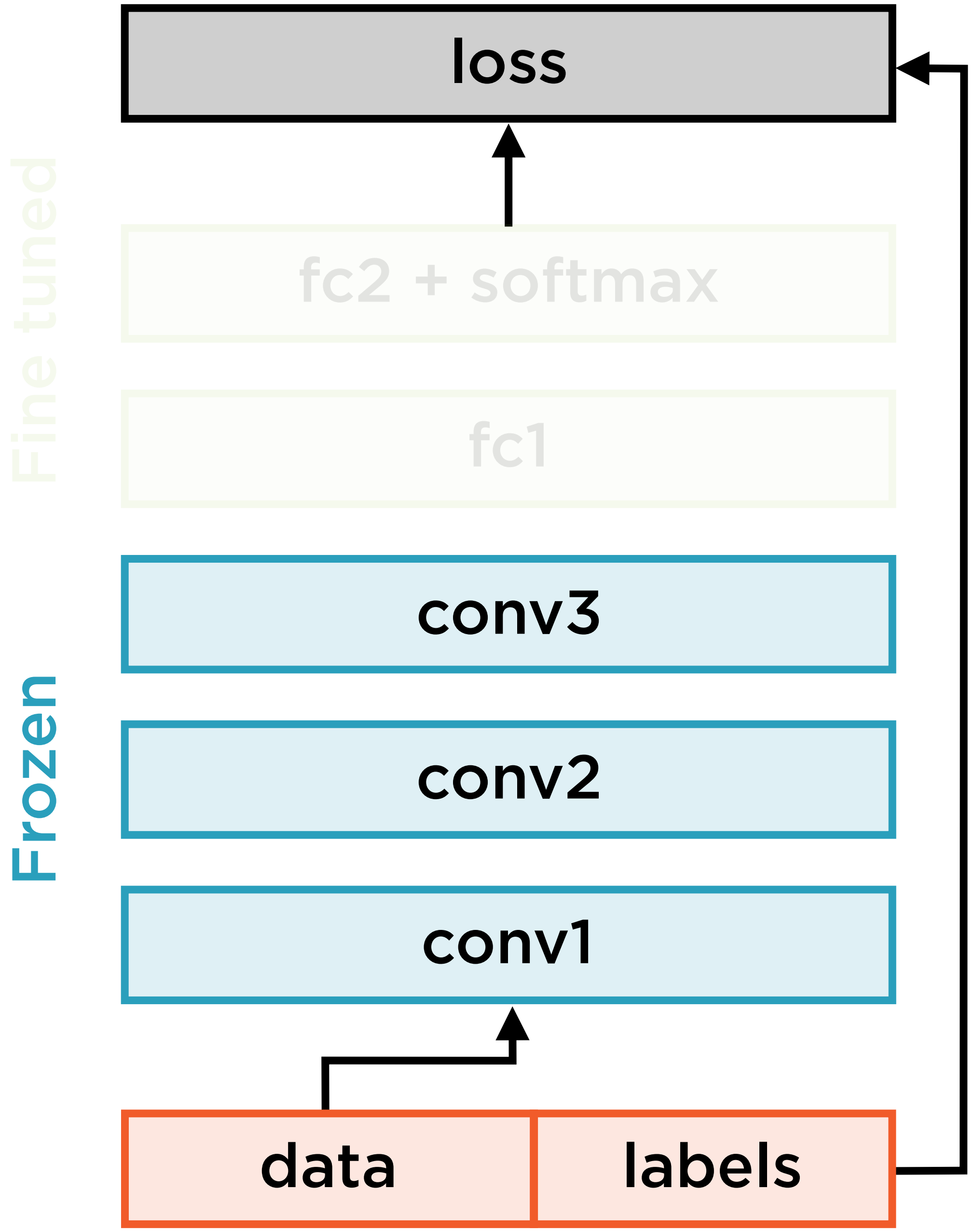
Freeze or Fine-tune?



Top-layers Specific to the Problem



Bottom Layers: Freeze or Fine-tune?



Freeze or Fine-tune?

Initial n **layers can be frozen or fine tuned.**

- Frozen: not updated during training
- Fine-tuned: updated during training

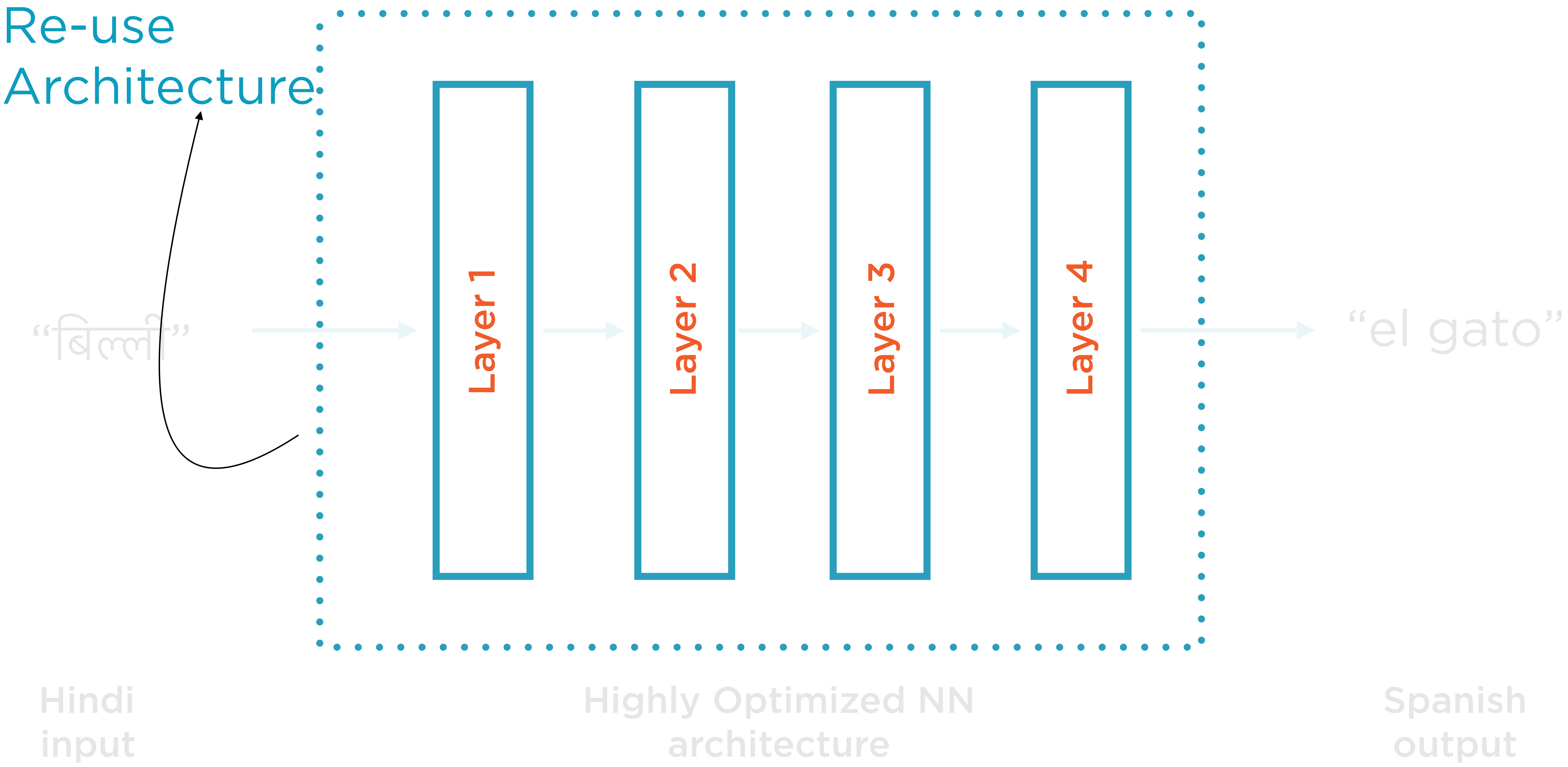
Freeze or Fine-tune?

Which to do depends on target task:

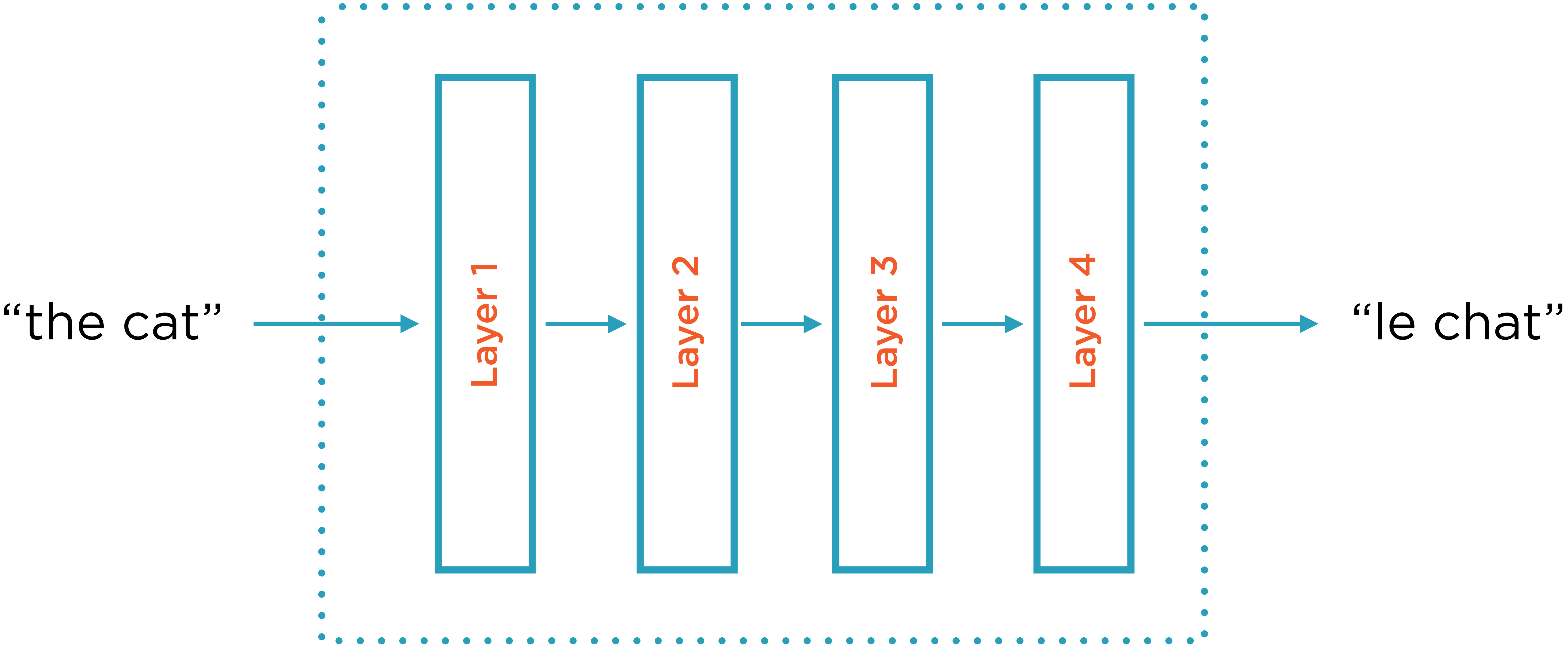
- Freeze: target task labels are scarce, and we want to avoid overfitting
- Fine-tune: target task labels are more plentiful

Can set learning rates to be different for each layer

Transfer Learning: Hindi to Spanish



Original Model: English to French

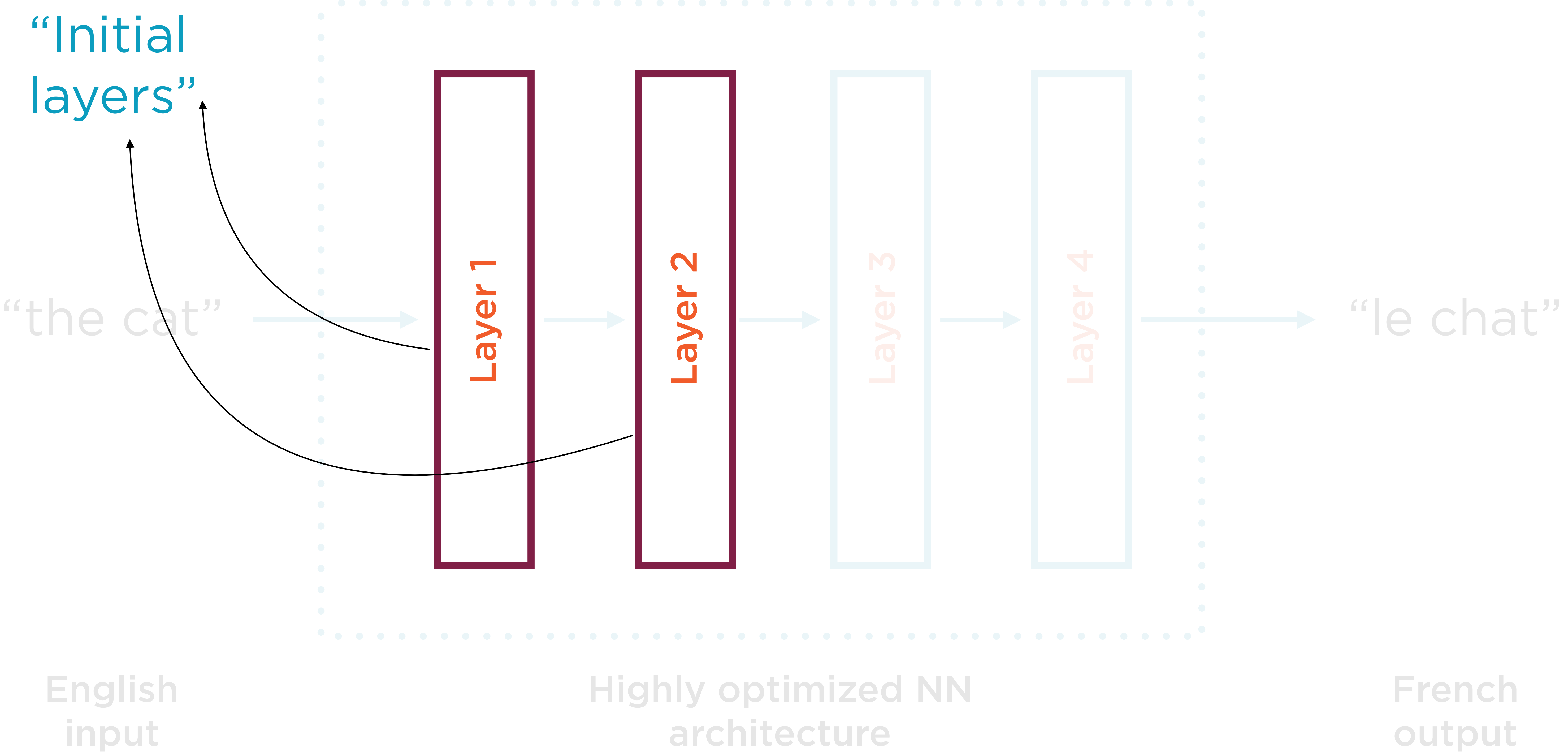


English input

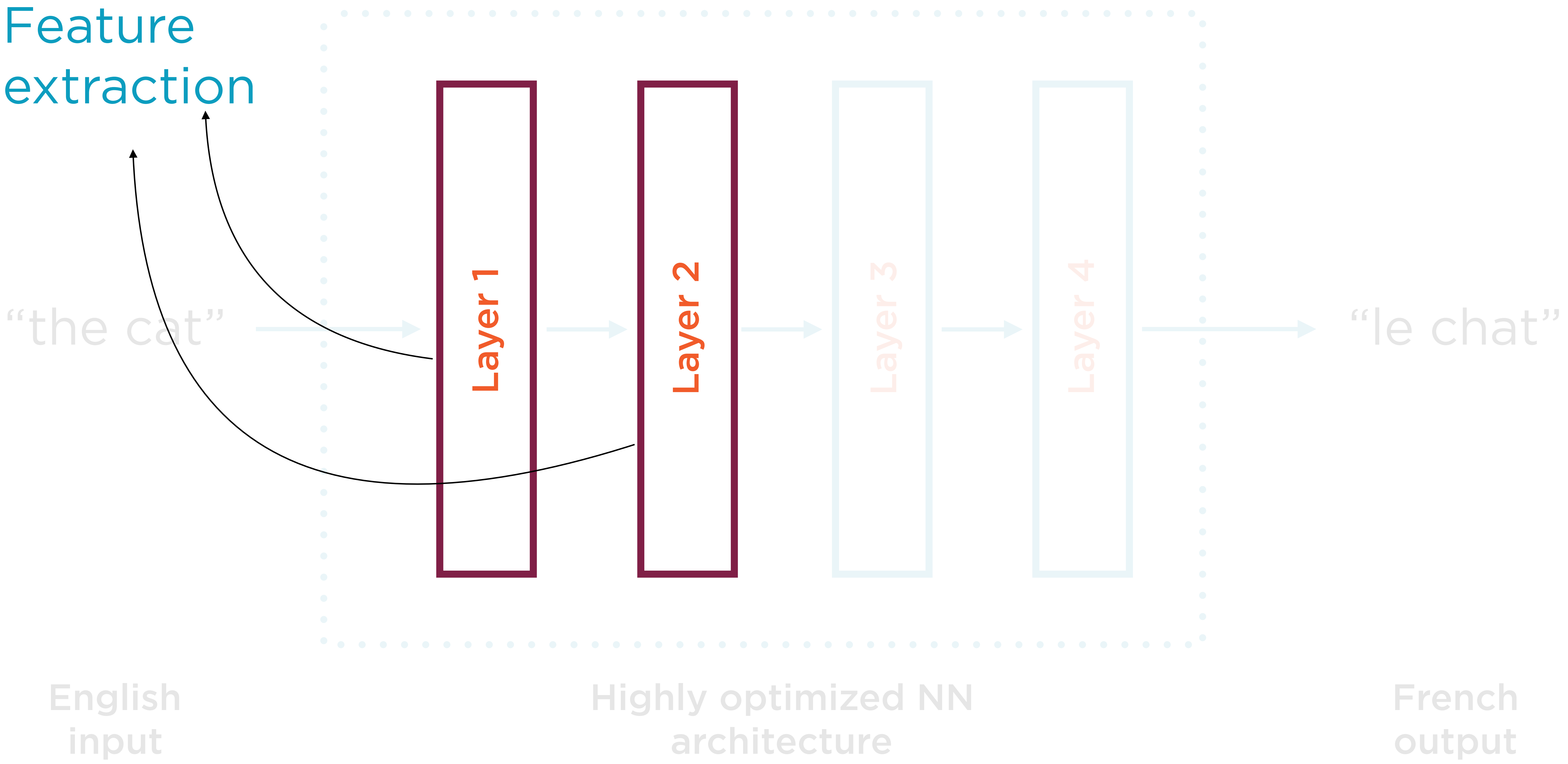
Highly optimized NN architecture

French output

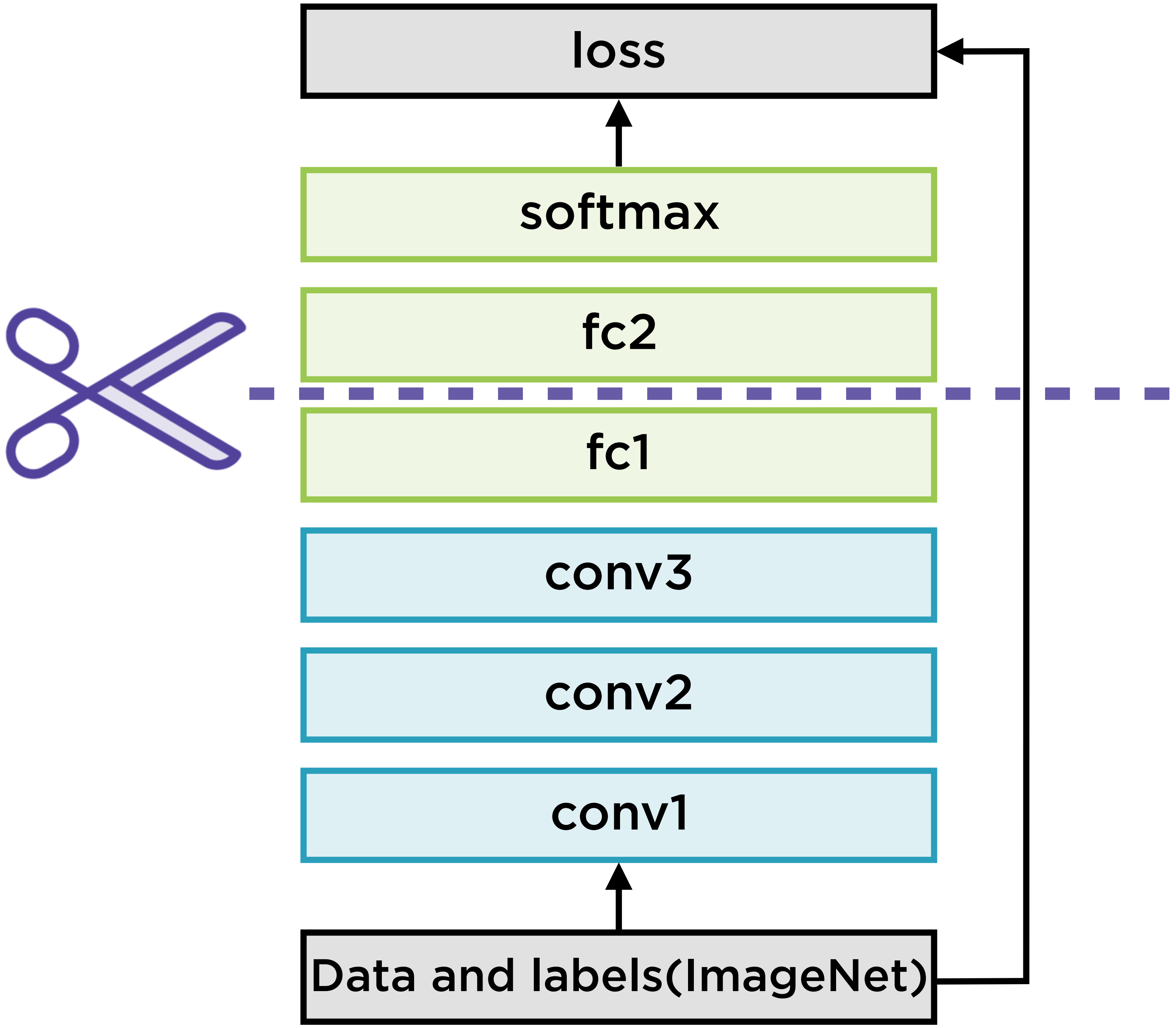
Original Model: English to French



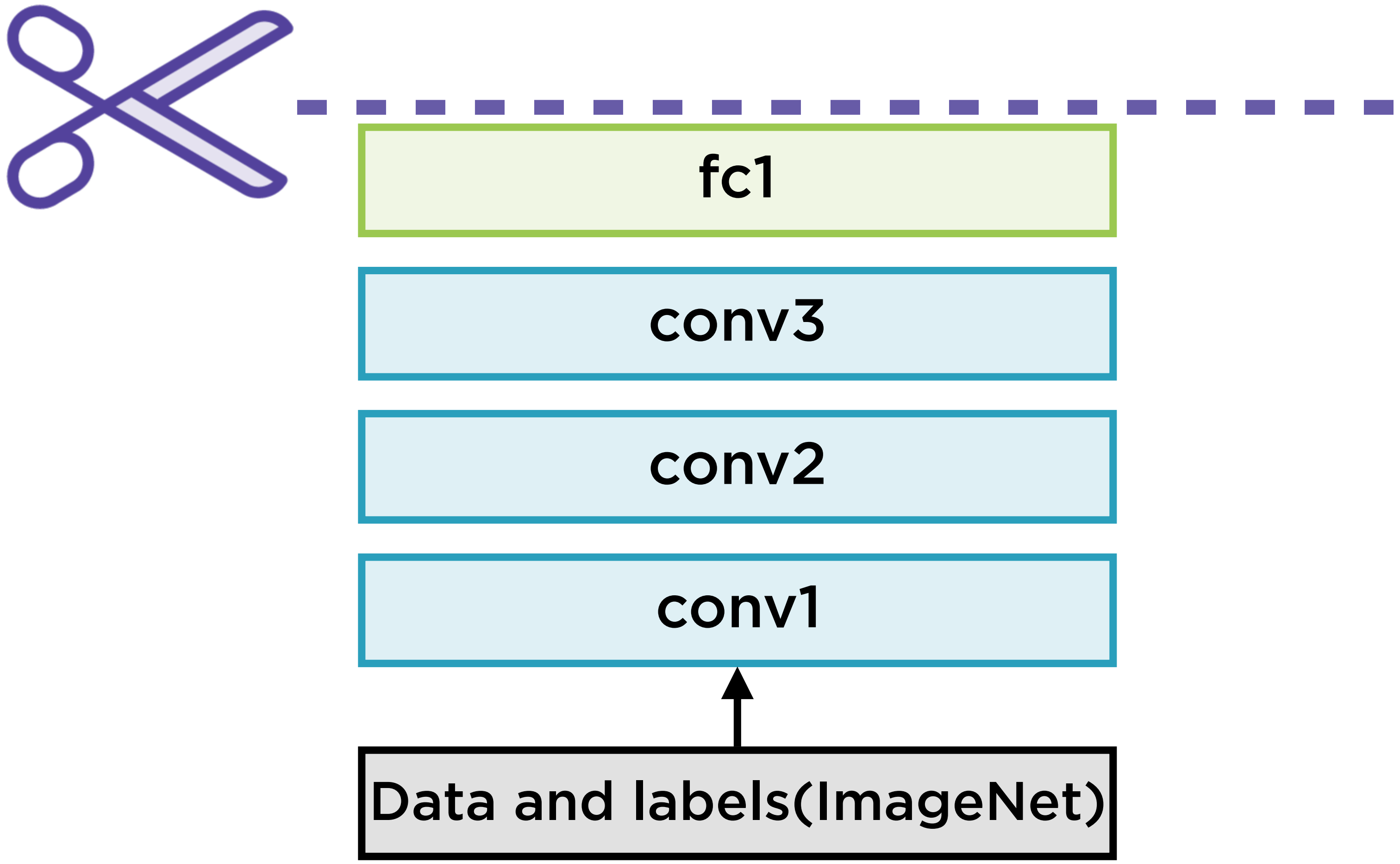
Original Model: English to French



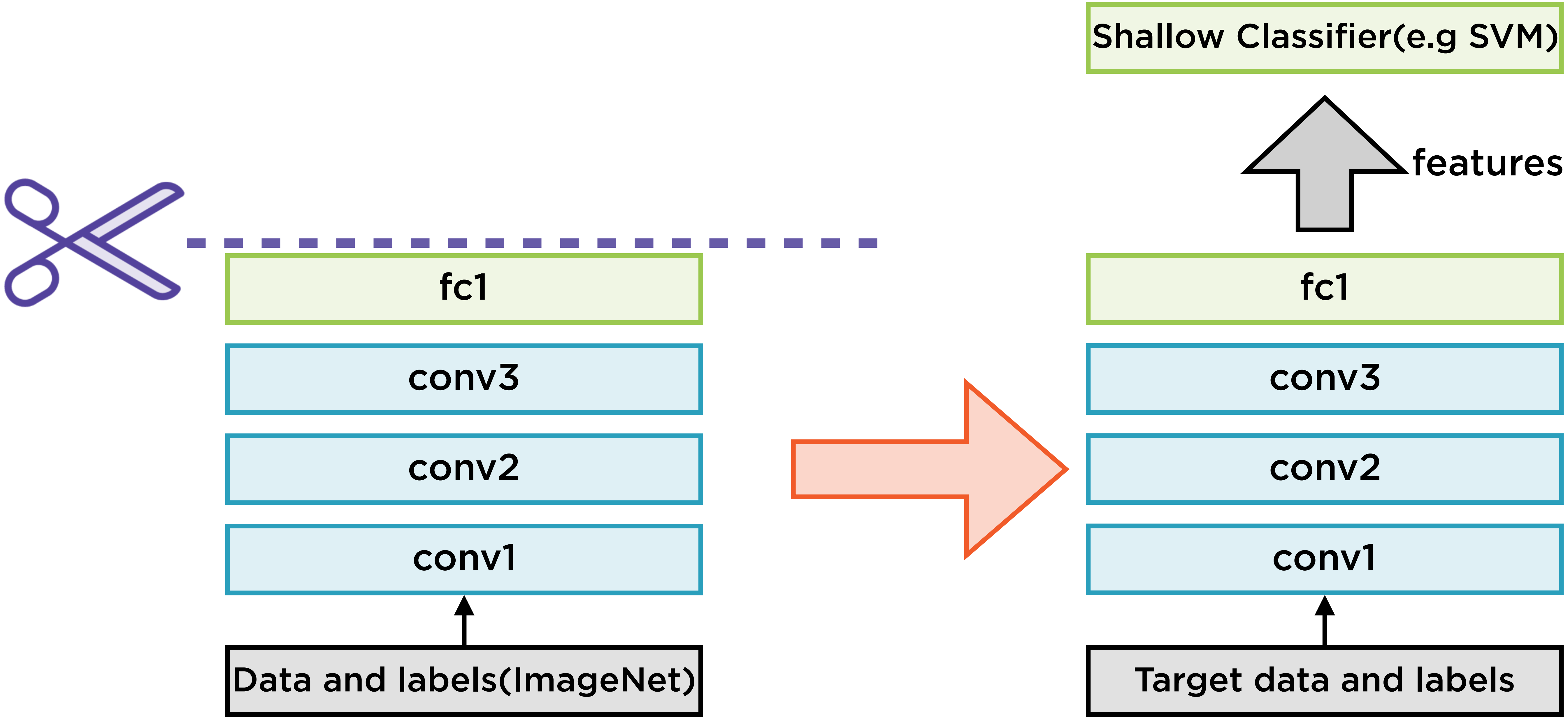
Initial Layers as Feature Extractors



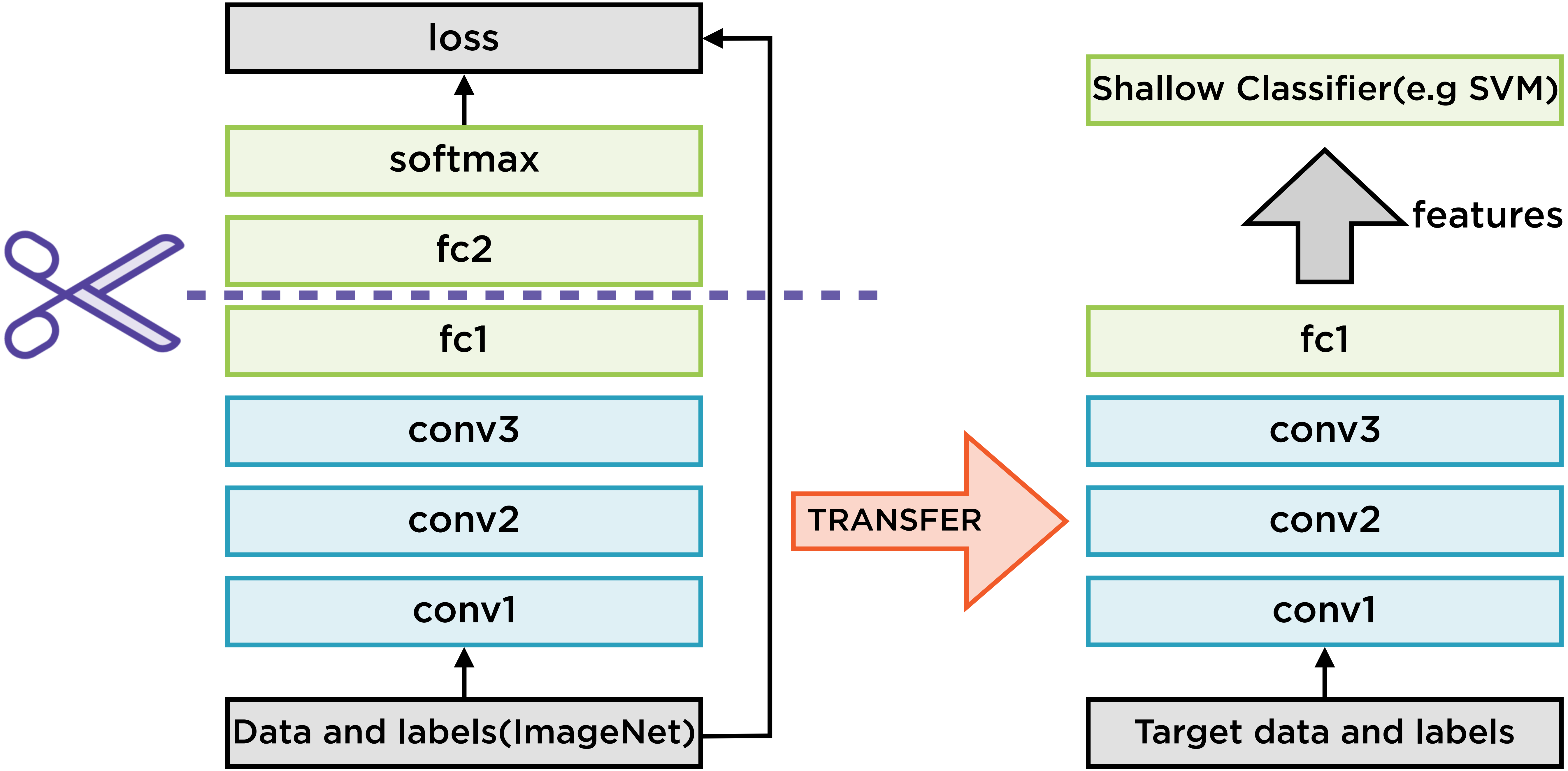
Initial Layers as Feature Extractors



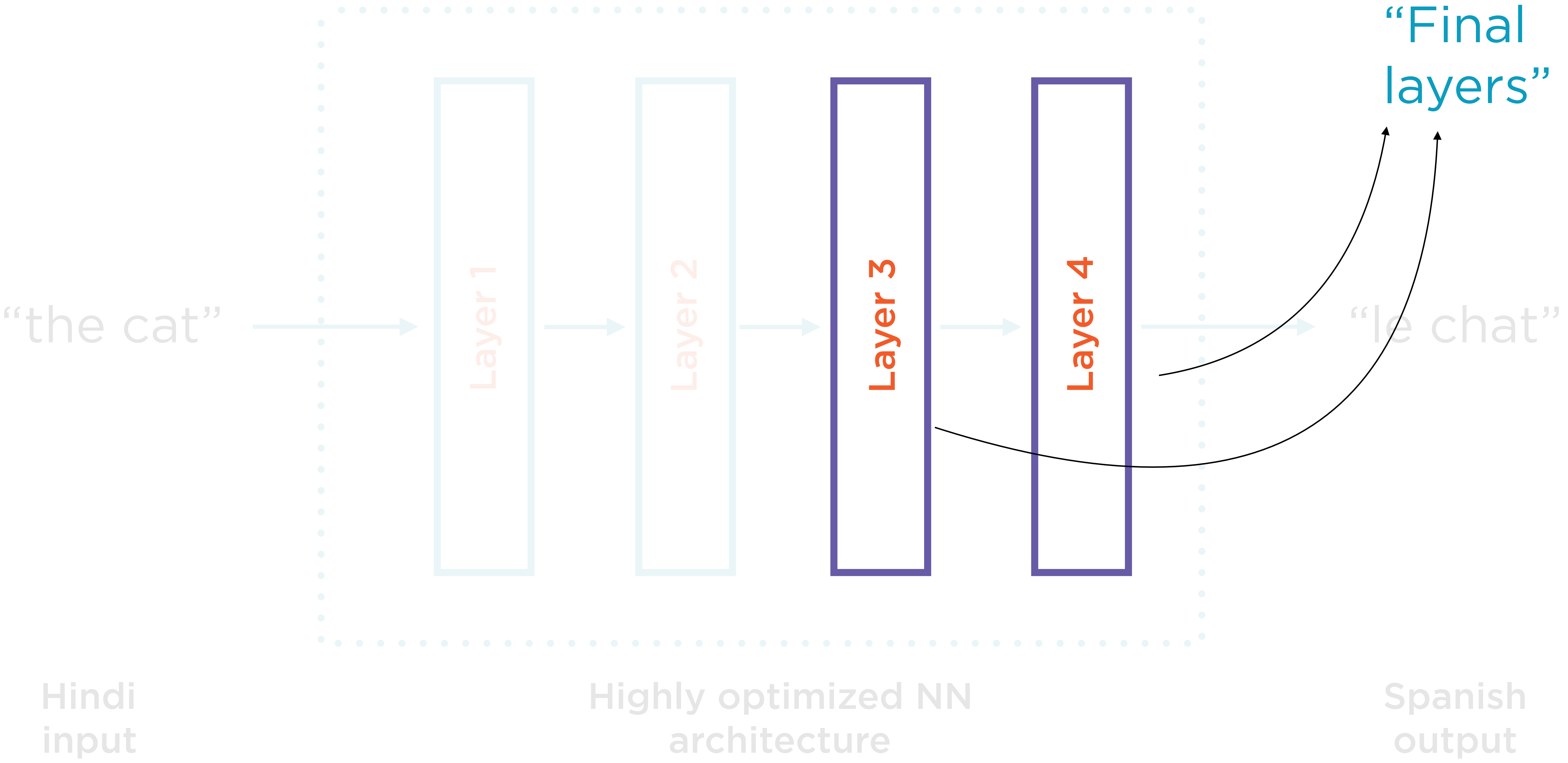
Initial Layers as Feature Extractors



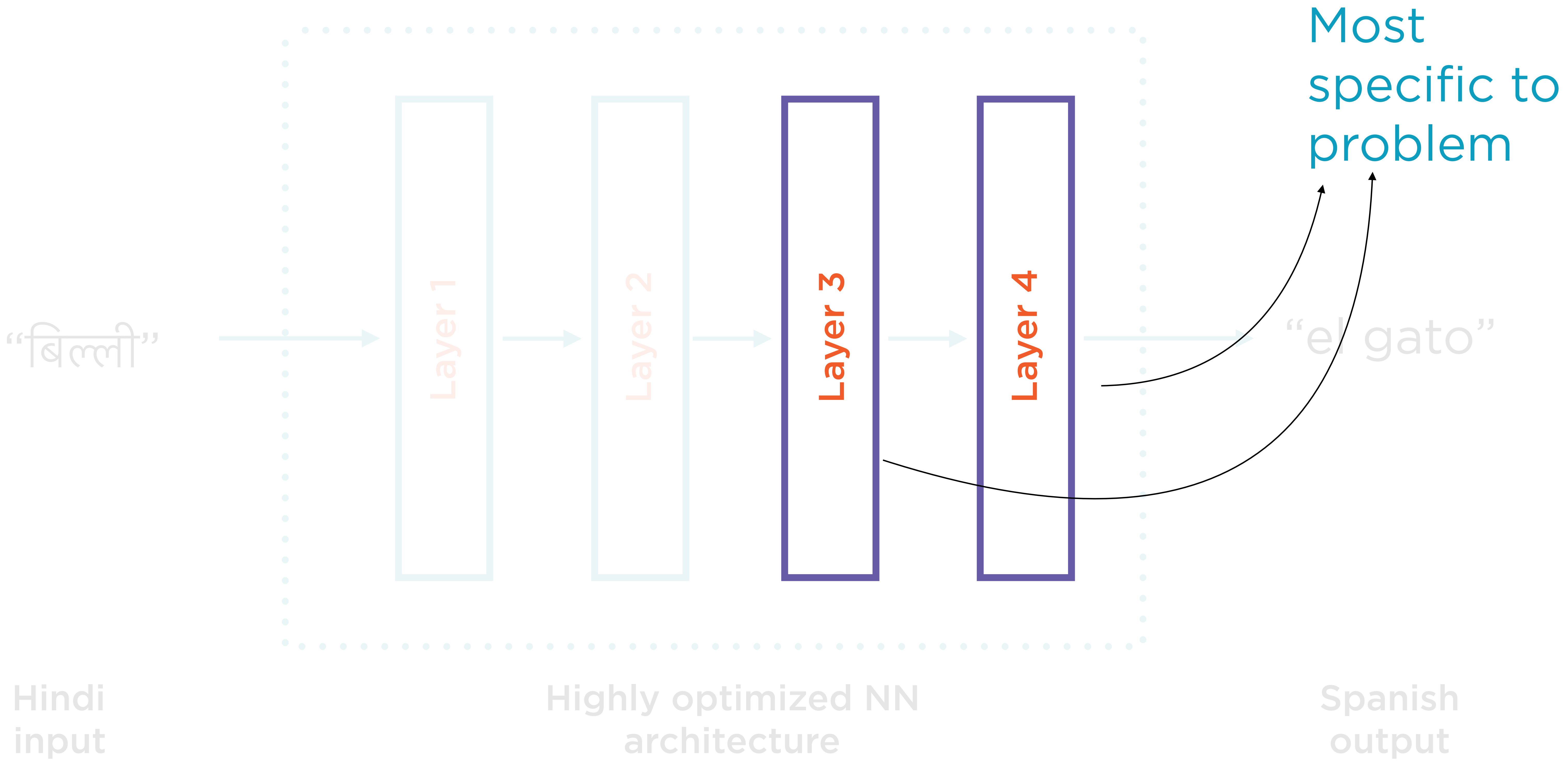
Initial Layers as Feature Extractors



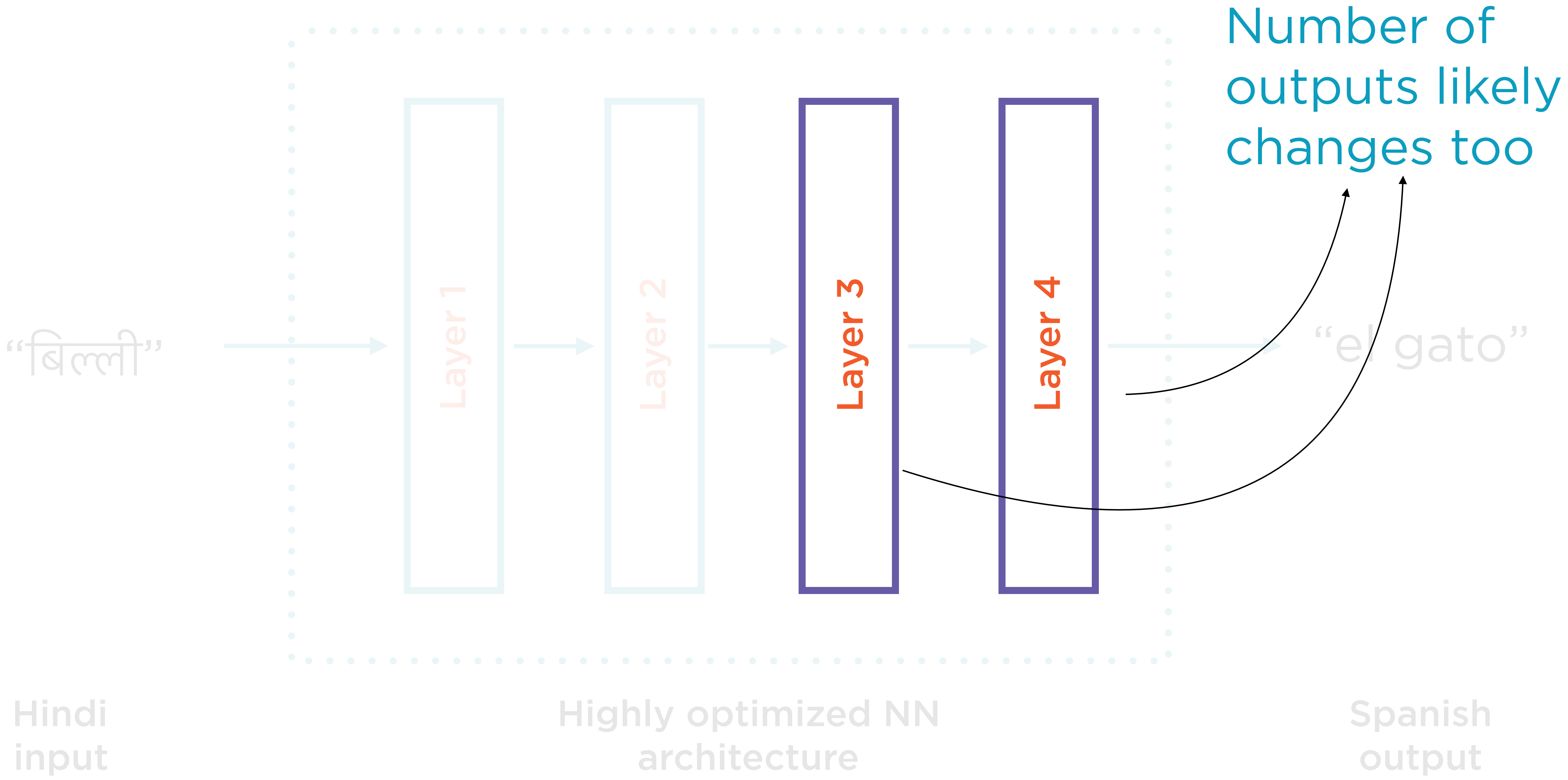
Original Model: English to French



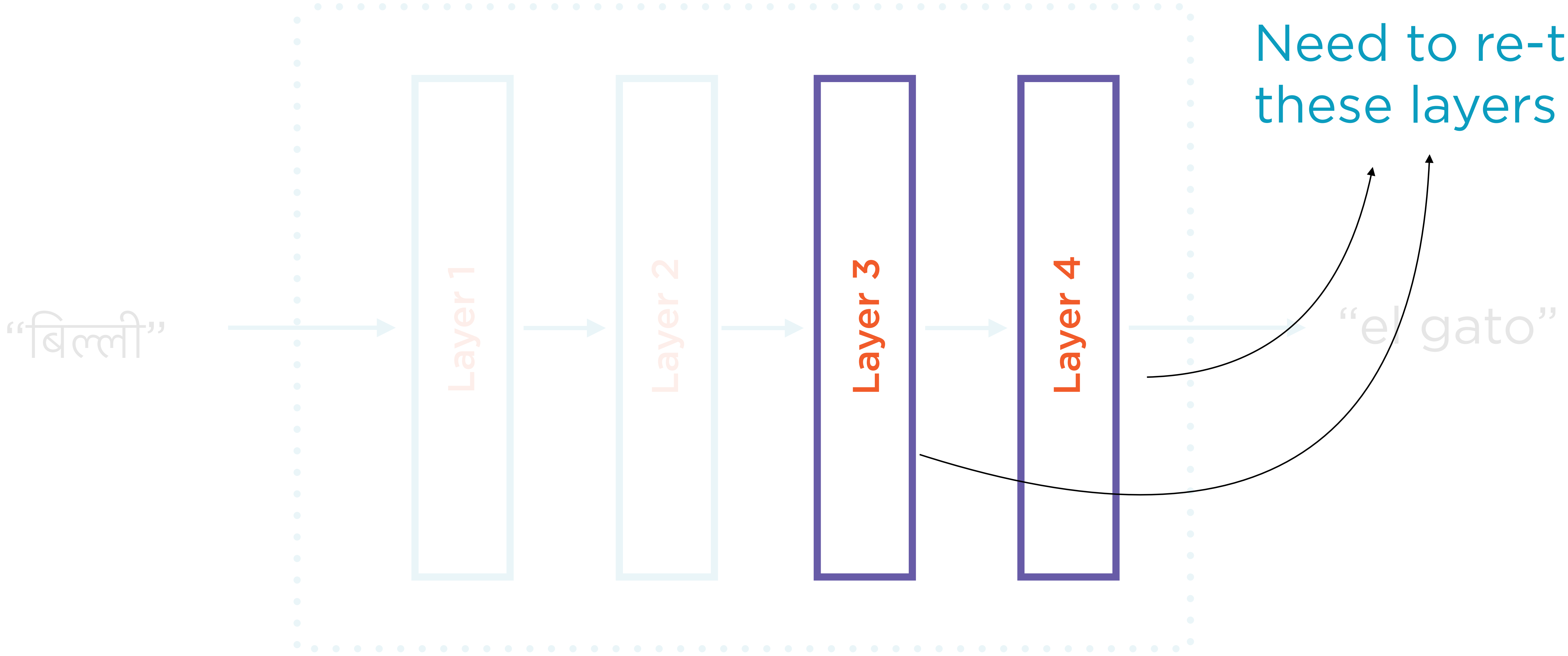
Transfer Learning: Hindi to Spanish



Transfer Learning: Hindi to Spanish



Transfer Learning: Hindi to Spanish



Need to re-train these layers

Hindi input

Highly optimized NN architecture

Spanish output

Benefits of Transfer Learning

Transfer Learning for Image Classification

Initial layers detect features common to all images

Color blobs, general filters, edges, lines

Later layers learn abstract details more specific to the problem

Benefits of Transfer Learning

“Ride on the shoulders of giants”

- NN architecture
- Choice of initialization
- Activation functions
- Number and density of layers

Benefits of Transfer Learning

“Do more with less”

Make do with less training data

- English to French: Lots of training data
- Hindi to Spanish: Little or no training data

Benefits of Transfer Learning

“Faster, cheaper”

Training process is far faster, easier

- Smaller training data
- Only higher layers to train
- In a cloud-enabled world, less time => less money

Transfer Learning in PyTorch

Transfer Learning in PyTorch

Support for several famous NN architectures

torchvision.models

- AlexNet
- VGG
- ResNet
- Densenet
- Inception and many others

PyTorch transfer learning models are trained on the ImageNet dataset

ImageNet

14 million images with 20,000 categories

Hand-annotated using crowdsourcing

Used for the famous annual contest

“ImageNet Large Scale Visual Recognition Challenge” (ILSVRC)

ImageNet

PyTorch models trained on a subset with 1000 categories

Transfer Learning in PyTorch

Support for several famous NN architectures

torchvision.models

- AlexNet
- VGG
- ResNet
- Densenet
- Inception and many others

Transfer Learning in PyTorch

Support for several famous NN architectures

torchvision.models

- **AlexNet**
- VGG
- ResNet
- Densenet
- Inception and many others

AlexNet

Big innovation - stack convolutional layers directly atop each other

Do not place pooling layers between these directly stacked layers

Mitigate overfitting risk by high dropout (50%) and randomly shifting training images by offsets

AlexNet

Uses form of normalization called “local response normalization”

Strongly activated neurons inhibit nearby neurons

Causes neurons to “compete” to specialize in different types of features

AlexNet won 2012 ImageNet contest by a huge margin

Transfer Learning in PyTorch

Support for several famous NN architectures

torchvision.models

- AlexNet
- VGG
- **ResNet**
- Densenet
- Inception and many others

ResNet

Famous CNN architecture

Won the ImageNet challenge in 2015

Extremely deep

“Skip connections” aka shortcut connections

Shares many features with typical CNN architectures

ResNet

Big innovation - “skip connections”

Connect output of lower layers to far-ahead higher layers

Batch normalization after each convolution and before each activation

Model is forced to focus on what is not learnt by intermediate layers

“Residual Learning”

Transfer Learning in PyTorch

Support for several famous NN architectures

torchvision.models

- AlexNet
- VGG
- ResNet
- **Densenet**
- Inception and many others

DenseNet

Extends idea of residual learning

Big innovation ~ Dense blocks, within which layers are densely connected to each other

DenseNet

Each dense block consists of layers with three components

- Batch normalization
- ReLU activation
- 3x3 convolution

DenseNet

DenseNet leads to compact models with relatively few parameters

Training is easy due to phenomenon called implicit deep supervision

Dense connections lead to gradient flowing back more easily

Transfer Learning in PyTorch

Support for several famous NN architectures

torchvision.models

- AlexNet
- **VGG**
- ResNet
- Densenet
- Inception and many others

VGG

Big innovation - stacking multiple small filters without pooling

E.g. Stack 3 convolutional layers of 3x3 rather than 1 convolutional layer of 7x7

Increase representational power without too many parameters

Small filters also provide regularization and mitigate overfitting

Demo

Set up a deep learning VM on the cloud

Demo

Explore pre-trained models available for image classification in PyTorch

Summary

Understand the use of pre-trained models and transfer learning

Understand source and destination domains

Understanding source and destination tasks

Learn when to use transfer learning

Explore PyTorch support for transfer learning